NEW DECODING ALGORITHMS FOR REED-MULLER AND CYCLIC CODES BASED ON SPECTRAL DOMAIN DECONVOLUTION

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Award of the Degree of
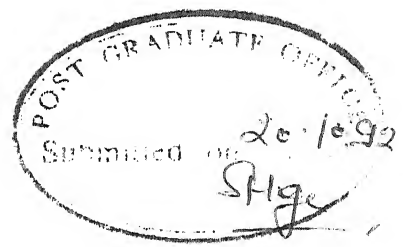
DOCTOR OF PHILOSOPHY
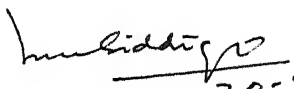
by

MADHUSUDHANA H S

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

October 1992

CERTIFICATE

It is certified that the work contained in the thesis entitled NEW DECODING ALGORITHMS FOR REED-MULLER AND CYCLIC CODES BASED ON SPECTRAL DOMAIN DECONVOLUTION by *Madhusudhana H S* has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.

M U SIDDIQI 20-10-92

Professor

Department of Electrical Engineering
Indian Institute of Technology Kanpur

October 1992.

EE-1992-M- MAD-NEW

|| ॐ ||

अपित्रिलोक्या बहिरुल्लसंती
तमोहरंती मुहुरान्तरंच ||
दिश्यात् दृशोनो विशदां जयंती
मध्वस्य कीर्तिर्दिननाथदीप्तिम् || ||

कुरु भुङ्क्ष्व च कर्म निजं नियतम्
हरिपाद विनम्रधिया सततम् ||
— श्री मध्वाचार्य:

मातापित्रन्तर्गत गुर्वन्तर्गत
श्री नारायण पादकमले समर्पणम् ||

ACKNOWLEDGEMENTS

v

It is a pleasure to thank my constant colleague and friend Udaya for his companionship and brotherly affection. I am greatly benefited from his comments and suggestions in my thesis work.

My thanks to Venkatesh and Deepak for suggestions and discussions at various stages of thesis work. My special thanks to Venkatesh for drawing ACAD diagrams at short notice and his assistance in the final stage of thesis preparation. I am thankful to Narayan, Jayatheertha and Aravind for their suggestions and assistance.

I thank all my friends, from seniors to juniors, with whom I have spent most of my time, for many fun filled moments, discussion on serious and non−serious matters of life and so many other things.

I express my appreciation to Messrs. Chourasiya Xerox Center for excellent photocopying and Ahmed Book Center for neat binding of the book.

I owe a lot to my parents, brothers and sisters for their love and support. I thank them for patiently bearing my repeated assurances that thesis will be completed in no time.

SYNOPSIS

The main thrust of the thesis is to expand the range of codes decodable using spectral techniques. Two classes of spectral transforms chosen for study are discrete Fourier transform (DFT) and Walsh–Hadamard Transform (WHT), both important special cases of Generalized Walsh Hadamard Transforms. Using these classes of transforms decoding problems, in the spectral domain, are shown to be equivalent to appropriate deconvolution problems. Structural properties of the transforms and codes are used to obtain simple decoding (deconvolution) algorithms. WHT techniques are applied for decoding of Reed–Muller (RM) codes of order 2 and 3 whereas DFT techniques are applied for decoding of different classes of cyclic codes.

Design and implementation of good decoding algorithms has attracted lot of attention in the literature. Since decoding speed puts a restriction on the overall throughput and a simpler decoder architecture reduces the cost of decoding, smallest possible decoding delay and simplest possible decoder architecture are the two important objectives of a good decoding algorithm. Achieving these objectives of reduced delay and implementational simplicity depends primarily on the mathematical structure of the code. In some cases, the structural symmetries directly suggest a simple implementation scheme. Threshold decoding scheme for the class of Reed–Muller codes and Meggit decoder scheme for the class of cyclic codes may be seen as examples in this category. In other cases, it may be necessary to seek alternative representations of codes for obtaining good decoding algorithms. Appropriate spectral (transform) domain representations of codes provide good means for developing efficient decoding algorithms; resulting algorithms are referred to as spectral or transform domain decoding algorithms and constitute an important subset of algebraic decoding algorithms. As examples may be mentioned DFT domain decoding algorithms for BCH codes [1] and WHT domain decoding algorithm for first order RM codes [2].

To date the BCH decoding algorithm continues to be the most important algorithm among spectral decoding algorithms based on DFT domain techniques [1]. It is found that the BCH decoding algorithm can be easily modified for correcting both errors and erasures and also for decoding beyond BCH bound. The concept of BCH decoding is applied for decoding of other related class of codes such as alternant codes and Goppa codes. Another application of BCH decoding idea is in developing decoding algorithms for the class of 2–D BCH codes obtained by generalizing the concept of BCH codes to two dimensions. Recently, some attempts have been made to decode some non–BCH cyclic codes using spectral methods [3]. A careful examination of different DFT based spectral decoding algorithms reveals certain basic principles which can be used to derive all these decoding algorithms.

The range of application of WHT techniques is limited to Reed–Muller codes. Presently decoding algorithms based on WHT methods are available only for first order RM codes. The attractiveness of WHT is that computations involve integer arithmetic and that decoding operations are relatively simpler. Taking the clue from the DFT based approach, it is worthwhile to look for some common principle from which one can derive WHT based decoding algorithms for higher order RM codes also.

It turns out that, from the system theoretic viewpoint, decoding of RM and cyclic codes can be viewed as appropriate deconvolution problems in WHT and DFT transform domains respectively. Therefore the aim is to look for simple ways of performing deconvolution using the structural properties of the codes and the relevant transforms.

The results obtained and the underlying approaches are summarized in the following pages.

1 WHT based Decoding Algorithms

A mapping from binary $\{0, 1\}$ to real $\{1, -1\}$ is necessary in order to apply WHT techniques. Then the EX–OR addition in binary field is mapped into pointwise multiplication in the real field and therefore the received signal in the time domain can be considered as the pointwise multiplication of error and the transmitted codeword. In the transform domain the received spectrum can be considered as dyadic convolution of the error spectrum and the transmitted codeword spectrum. The error spectrum can accordingly be visualized as the impulse response of a linear filter which convolutionally distorts the transmitted codeword spectrum. The decoding problem can be formulated as deconvolution of the WHT of received signal to recover the codeword spectrum. It is found that WHT spectrum of RM codes can be grouped into several histogram classes, where a histogram class consists of all transformed RM code vectors having the same distribution of spectral values and their frequency (For histogram only absolute values are considered). It is shown that the maximum possible deviation of the WHT spectrum of the received signal from the WHT spectrum of the codeword is linearly related to number of errors occurred. This information is useful to derive simple procedures for determining the histogram class of the codeword spectrum and consequently to recover the transmitted codeword from that of the received word.

Let $R(r,2^m)$ denote r^{th} order RM code of length 2^m. The WHT spectrum of $R(1,2^m)$ is known to have only one nonzero component with a value equal to $\pm 2^m$. Therefore $R(1,2^m)$ has a single histogram class consisting of a single component with the absolute value 2^m and the remaining 2^m-1 zero valued components. Our contribution is in extending WHT spectrum based histogram characterization to $R(2,2^m)$. We show that the WHT spectrum of $R(2,2^m)$ has $\lfloor m/2+1 \rfloor$ different histogram classes, j^{th} class containing 2^{2j} nonzero spectral locations having value $\pm 2^{m-j}$ for $j = 0$ to $\lfloor m/2 \rfloor$. Proof of various results are based on automorphism group property of RM codes and a theorem concerning canonical representation of multivariable polynomials over a binary field. Unfortunately, histogram characterization

could not be extended to third and higher order RM codes primarily because of non—availability of appropriate canonical forms.

Our next contribution is in using the histogram characterization of RM codes for their decoding. To proceed with the decoding we need to know how the histogram of the WHT spectrum of the code is modified due to channel errors. This computation is in general quite tedious except in case of small errors. Fortunately, it is found that the knowledge of range of variation of spectral values is useful for decoding purpose. The maximum possible deviation of a spectral value is shown to be linearly related to number of errors occurred.

The possibility of recovering the histogram of the transmitted codeword spectrum from that of the received word spectrum is based on the following two conditions: first, the sign of a nonzero spectrum should not change due to error modification and second, different spectral values within a spectral class are far enough so that their range of variation are distinct. An analysis of these conditions shows that it is still not possible to decode $R(2,2^m)$ in a straightforward manner, the exception being $R(2,2^4)$, which is the first nontrivial single—error correcting second order RM code of length 16. We give a WHT decoding algorithm for single—error correcting $R(2,2^4)$ code and utilize it to develop a decoding algorithm for general $R(2,2^m)$.

In order to obtain WHT domain decoding algorithm for general $R(2,2^m)$, we utilize the superimposition property [2] of a RM code to decompose a larger RM code into smaller RM codes having efficient decoding algorithms, which can be combined to provide a decoding algorithm correcting upto full error correcting capability for the larger code. In [4] Tokiwa et.al. have given an algorithm (which we refer to as TSKN algorithm) based on the superimposition property which reduces the larger RM decoding into several smaller single—error—correcting and double—error—detecting RM codes having an efficient decoding algorithm. They have also shown that TSKN algorithm has better delay compared to conventional majority logic decoding algorithms for RM codes.

The decoding algorithm given in this thesis is based on a decomposition that reduces decoding of a $R(2,2^m)$ to $(m-4)$ fold decoding of $R(1,2^{m-1})$ and a single decoding of $R(2,2^4)$ repeated 2^{m-4} times (For brevity the repeated code is represented as $2^{m-4}*R(2,2^4)$). A new WHT based decoding algorithm for decoding of $2^{m-4}*R(2,2^4)$ is developed on the lines of WHT based decoding algorithm for $R(2,2^4)$. This new WHT based algorithm and the well—known WHT based decoding algorithm for $R(1,2^m)$ are combined to provide decoding algorithm for $R(2,2^m)$. We propose two variations of the decoding algorithm for $R(2,2^m)$, one of which avoids the need for inverse WHT and also simplifies the process of recovering information bits.

We show in a similar way that decoding $R(3,2^m)$ reduces to decoding of $(m-5)$ fold $R(2,2^{m-1})$ and a single decoding of $2^{m-5}*R(3,2^5)$. WHT domain decoding algorithms for $R(3,2^5)$ and $2^{m-5}*R(3,2^5)$ are given and utilized for developing a decoding algorithm for $R(3,2^m)$ by combining decoding algorithms for $2^{m-5}*R(3,2^5)$ and $R(2,2^m)$.

There are important differences in the workings of the TSKN decoding algorithm and the WHT based algorithm, apart from different ways of decomposing a RM code. Both the methods at some stage have to do decoding of repeated codes. TSKN method attempts to decode each code block of the repeated code and then ascertains the validity of decoding. If the result is not valid, then it takes the next code block and repeats the process. The number of attempts varies from one to as many as the number of repeated code blocks. WHT based method attempts to decode the entire repeated code in a single step. In case there are ambiguities regarding the spectral pattern of the code, it requires one more step to complete the decoding. Therefore the number of attempts is at most 2. This feature may simplify the decoder architecture compared to TSKN method. We give a decoder architecture based on the WHT domain decoding algorithms for second and third order RM codes. An estimate of the decoding delay with this architecture is given. The delay is shown to be significantly less compared to TSKN algorithm, which in turn has smaller delay compared to other decoding algorithms for higher order RM codes.

2 DFT based Decoding Algorithms

Both 1–D and 2–D DFT based algorithms depend upon characterization of cyclic codes as having a specified set of codeword spectral components identically equal to zero. As a consequence the DFT spectrum of the received vector at these locations is equal to the value of the error spectrum. This direct availability of partial information on error spectrum is in contrast to its non–availability in case of WHT. By introducing an auxiliary vector which is zero at locations corresponding to error locations, considering the pointwise product of the error vector and the auxiliary vector, and transforming the pointwise product, a "zero" output convolution involving the error spectrum is formed. Therefore DFT domain decoding can be formulated as zero–output deconvolution with partial knowledge of the error spectrum. The other convolving polynomial, DFT of the auxiliary vector introduced earlier, is also of importance in that its zeros are related to the locations of the error and hence known as the error–locator polynomial. In the DFT domain the deconvolution takes the form of problem of finding the connection polynomial of least order LFSR generating the entire error spectrum with the known error spectrum at the zero locations of the code as the initial values of the LFSR. In view of this the error–locator polynomial is also referred to as the connection polynomial.

We treat 1–D and 2–D decoding algorithms in a single framework. Analysis of a typical DFT domain decoding algorithm reveals three important stages: the error spectrum computation stage, the error location stage, and the error evaluation stage. Any DFT based decoding approach centers around the error location stage which involves formulation and computation of error locator polynomials. Any particular decoding approach is typified by the nature of the deconvolution corresponding to the nature of the convolution of the transform domain vectors of the corresponding error locator and error spectrum polynomials. For efficient computation, the error location stage requires some structure on the zero

locations of the code; this also influences the error evaluation stage. Error evaluation can be completed in various ways and an appropriate choice is made so as to reduce the overall decoding complexity.

We study two distinct DFT decoding approaches: 2–D deconvolution approach based on Sakata's algorithm for 2–D LFSR synthesis [5] and 1–D deconvolution approach based on a 2–D linear complexity theorem (CT). The latter approach is our contribution based on a 2–D extension of Blahut's 1–D linear complexity theorem [6].

In the 1–D BCH decoding the concept of an error locator is to associate a position 'i' with α^i, where α is the primitive n–th root of unity (n is the length of the code). The basic idea of generalization from the standard 1–D BCH case is the association of a nonzero error position (i,j) with a 2–D error–locator (α^i, β^j) corresponding to row and column indices of the error position. Then the 2–D convolution of the transform domain vectors of the corresponding error–locator and error spectrum polynomials is zero and therefore the decoding problem is formulated as a 2–D DFT zero–output deconvolution. The error–locator polynomials or recurrence polynomials can be interpreted as two variable polynomials. This interpretation immediately suggests that all polynomials that satisfy a given array of 2–D error spectrum constitute an ideal. Then the problem of LFSR synthesis is shown to be equivalent to finding a "distinguished basis" of the ideal which is minimal in the sense of some total ordering for a 2–D array. Sakata has given an algorithm for efficiently computing such a basis which is equivalent to finding an algorithm for 2–D LFSR synthesis [5]. The error–evaluation stage consists of either solving for the roots of the system of connection polynomials for individual error positions in the binary case or performing 2–D recursive extension of the known error spectrum to obtain the complete error spectrum.

We propose an alternative approach to obtain DFT based spectral decoding algorithms centered around a generalization of Blahut's 1–D complexity theorem for a linear recurring sequence [6]. The complexity of a sequence is defined as the minimal order of LFSR generating the sequence. This notion of complexity has proved to be very useful in deriving the minimum distance bounds and in determining the weight structure of codes. We show that it can also play a crucial role in the formulation and design of decoding algorithms. This fact is more evident in the 2–D decoding algorithms.

Blahut's 1–D complexity theorem states that the minimal order of LFSR generating a time (or frequency) domain sequence is equal to the Hamming weight of transform sequence in frequency (time) domain. The roots of the connection polynomial correspond to locators of the nonzero components of the time domain vector. We show that the notion of linear complexity is easily generalized to a 2–D sequence as follows. The minimal order of common LFSR generating all row (column) subsequences is equal to number of nonzero columns (rows) in the transform sequence. The roots of the common row (column) connection polynomial gives the columns (rows) corrupted by the error.

The basic idea of our approach is to associate a nonzero error position (i,j) with a 1–D error–locator corresponding to either row (α^i) or column (β^j) index of the error position. Then the 1–D convolution o

row (column) error locator vector with each of row (column) subsequences of the 2—D error spectrum is zero. A typical 2—D decoding problem is then viewed as equivalent to solving several 1—D deconvolution problems which for the DFT case are equivalent to synthesizing several 1—D LFSRs corresponding to the row or column subsequences. Note that in contrast to 2—D deconvolution approach, LFSR synthesis is treated as 1—D whereas the error spectrum is 2—D in nature. In the error evaluation stage the recursive extension has to be done in both the column and row directions. It is also easy to see that the 2—D span of locations corresponding to product of zeros of the common row and column connection polynomials contains all the error positions. This makes 1—D deconvolution approach more elegant to take care of 2—D burst—errors compared to 2—D deconvolution approach. As the 1—D deconvolution approach is based on 1—D LFSR synthesis, this may imply milder constraints on the structure of zero locations, for a given error—correcting capability, compared to 2—D deconvolutional approach which is based on Sakata's algorithm for 2—D LFSR synthesis. A decoding example is given to illustrate this advantage of 1—D deconvolutional approach over 2—D deconvolutional approach.

For a better understanding of various DFT decoding algorithms, the relationship between error—locator polynomials that arise in 2—D and 1—D deconvolution methods is of interest. A look at the ideal structure of error—locator polynomials reveals that the ideal formed in the latter case is a sub—ideal of ideal formed in the former case. This is because in the former approach we are interested in individual error locations whereas in the latter approach we are interested in the span of error locations. Therefore in the 2—D deconvolution method the error—locator polynomials can be algebraically solved for individual error locations whereas in the 1—D deconvolution method solution of the equations do not give exact locations of errors but only span of error locations. An algorithm due to Fitzpatrick et. al. in [7] is invoked to show how to obtain the set of error—locator polynomials in 2—D deconvolutional approach using the error—locator polynomials obtained in the 1—D deconvolutional approach. We note that this indirect way to computing the error—locator polynomials is not efficient from the decoding viewpoint. However, this establishes the theoretical connection between these two approaches and one can continue in either approach to obtain valuable insights. For reasons explained in the above paragraph, we have preferred to use 1—D deconvolution approach.

a) DFT based Spectral Decoding Algorithms for 2—D BCH Codes

We begin with applications of 1—D deconvolutional approach to look into decoding algorithms for the class of 2—D BCH codes which are defined as having a contiguous array of 2t x 2t zeros. These codes have simple structure that helps in understanding the basic principles of 2—D decoding. Blahut has given a decoding algorithm for random errors correcting upto minimum distance bound of 2t+1 [1]. He has also given a decoding algorithm for correction of some patterns of 2—D burst errors [1].

We take a different approach to derive Blahut's algorithm. This approach provides a better insight into the working of the decoding algorithm. The inter–relationship between polynomials are viewed from the point of view of the linear recurrence theory. This shows that certain computations can be cut down compared to Blahut's decoding method. In particular, it is shown that at most "t" passes through B–M algorithm are required to find the connection polynomial as opposed to "2t" or more in Blahut's case. An "Improved Blahut's algorithm" for random error–correction incorporating the above mentioned modifications is presented. The improvements in algorithms can also be used to advantage for decoding for burst–error correction. Another significant question in the context of 2–D BCH codes is that of maximum correction capability of structure of zero locations in the form of 2t x 2t contiguous array. We employ 2–D complexity theorem to establish an error–correcting limit that the structure of zero locations allows. It follows as a result of these investigations that Blahut's algorithm is best possible in that it corrects upto the limit. Next we introduce a class of Γ 2–D BCH codes that has less number of zeros compared to the class of 2–D BCH codes but has the same error–correcting capabilities. We employ 2–D complexity theorem to show that the error–correcting limit cannot be further improved. As it is not possible to apply Blahut's decoding algorithm directly, we give a decoding algorithm to correct upto the limit as shown by complexity theorem arguments. A time domain implementation of Blahut's original as well as modified decoding algorithms is also given. We also discuss an alternative approach to decode the class of 2–D BCH codes on the basis of multiple LFSR synthesis algorithm proposed by Tzeng and Feng [8].

Next we consider a class of 2–D BCH codes from the perspective of only random error correction and give a decoding algorithm for these codes. This study intuitively helps in evaluating the requirements on the structure of the zero locations made by different types of random and burst errors.

b) DFT based Spectral Decoding Algorithms for 2–D Cyclic Codes

We next apply the 1–D deconvolutional approach to give decoding algorithms for non–BCH cyclic codes. These are not as structured as BCH codes. Nevertheless their performance is comparable to BCH codes. Though some attempts to develop decoding algorithm for such codes have been made, so far there are no decoding algorithms that apply to a larger class of codes [3]. We hope that the given 1–D deconvolutional approach provides basic principles which can be tailored for decoding of specific codes.

The main idea in developing the decoding algorithms is to analyze the changes in the degree of the connection polynomials, due to different configurations of error patterns of a given weight, upto some limit which should desirably coincide with the correcting limit $\lfloor (d_{min} -1)/2 \rfloor$ of the code. The decoding algorithms proceed in stages anticipating various possible error configurations, finding appropriate connection polynomials, and testing for their validity. Validity tests includes periodicity tests on error spectrum, testing for conjugacy constraints and checking for consistency of row (column) connection

polynomials with known error spectrum along that row (column). These steps naturally lead to a relatively more complex decoding algorithm.

The main effort is how to find the appropriate row or column connection polynomial for a given configuration of errors using all possible information about the error spectrum. This is motivated by the engineering considerations and helps in a better understanding of the algorithm. The approach may be seen to carefully balance the requirements of algebraic complexity and implementation complexity.

Some of the novel features of the 2–D spectral decoding algorithms for non–BCH cyclic codes are as follows:

1) The algorithms sometime allow interesting variations of Chien search methods and trial–and–error methods.

2) The algorithms can be modified for erasure correction on a limited scale on the lines of standard BCH and Feng–Tzeng multi–sequence shift–register synthesis algorithms [8].

3) The algorithms are particularly suited for correction of periodic erasure bursts.

We have given 2–D decoding algorithms for both 1–D cyclic and 2–D cyclic codes. The decoding algorithm for 1–D cyclic codes are based on the 2–D cyclic characterization of 1–D codes. We have chosen several 1–D cyclic codes of lengths 21, 33, 35, 39, 45, 51 and 63 from those complied in [9]. Decoding algorithms for these codes exhibit different techniques and all of them decode upto the error–correcting capability of codes. For some of these codes an algebraic decoding algorithm has been proposed for the first time.

Some good 2–D concatenated cyclic codes have been compiled by Jorn Jensen [10]. Decoding methods for these codes make use of concatenated property of codes. We consider almost all good codes of length 45,49 and 75 given in [10] and give decoding algorithms, some of them being good ones, based on our approach, without needing details of the components of the concatenated code.

We have showed that decoding problems in both the DFT and WHT domain are equivalent to appropriate deconvolution problems and used transform domain properties for obtaining decoding (deconvolution) algorithms. It is noted that WHT based method directly gives the codeword spectrum estimate whereas DFT based methods give error estimate directly. Another distinctive feature of the WHT approach to decoding is in using histogram properties of WHT spectral coefficients to separate the features of error and codeword whereas in DFT based approach the separation is obtained based on the characterization of a code having specified DFT coefficients identically equal to zero. We next formulated methodologies for decoding and applied them to obtain decoding algorithms for RM codes of order 2 and 3 and various classes of cyclic codes. Although not intended as part of our basic approach, the spectral decoding algorithms also lead to decoder implementation schemes having reduced decoding delays and simpler decoder architectures.

REFERENCES

1) R. E. Blahut, "Theory and Practice of Error Control Codes", Reading, MA: Addison–Wesley Publishing Company, California, 1983

2) F. J. MacWilliams and N. J. A. Sloane, "Theory of Error–Correcting Codes," North–Holland Publishing Company, New York, 1981

3) P. Bours, J. C. M. Janssen, M. van Asperdt and H. C. A. van Tilborg, " Algebraic decoding beyond e_{BCH} of some binary cyclic codes, when $e > e_{BCH}$", *IEEE Trans. on Inform. Theory*, VOL IT–36, No. 1, pp 214–222, Jan. 1990

4) K. Tokiwa, T. Sugimura, M. Kasahara and T. Namekawa, "New Decoding Algorithms for Reed–Muller Codes," *IEEE Trans. on. Inform. Theory*, VOL IT–28, No. 5, pp 779–787, Sept. 1982

5) S. Sakata, "Finding a Minimal Set of Linear Recurring Relations Capable of Generating a Given Finite Two–Dimensional Array," J. Symbolic Computation, No. 5, pp 321–337, 1988

6) J. L. Massey and T. Schaub, "Linear Complexity in Coding Theory," in "Coding Theory and Applications," Ed. G. Cohen and P. G. Godelwski, Lect. Notes in Comp. Sci., No. 311, pp 19–32, 1988

7) P. Fitzpatrick and G. H. Norton, "Finding a Basis for the Characteristic Ideal of an n–dimensional Linear Recurring Sequence," *IEEE Trans. on Inform. Theory*, VOL IT–36, No. 6, pp 1480–1487, Nov. 1990

8) H. Shahri and K. K. Tzeng, "On Error–and–Erasure Decoding of Cyclic Codes," *IEEE Trans. on Inform. Theory*, VOl IT–38, No. 2, pp 489–496, May 1992

9) J. H. van Lint and R. M. Wilson, "On the Minimum Distance of Cyclic Codes," *IEEE Trans. on Inform. Theory*, VOl IT–32, No. 1, pp 23–40, Jan 1986

10) Jorn M. Jensen, "The Concatenated Structure of Cyclic and Abelian Codes," *IEEE Trans. on Inform. Theory*, VOL IT–31, No. 6, pp 788–793, Nov. 1985

TABLE OF CONTENTS

CHAPTER 6   SPECTRAL DOMAIN DECODING ALGORITHMS FOR 2–D CYCLIC CODES 167

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

$\lfloor m \rfloor$	:	Greatest integer $\leq$ m.
$\lceil m \rceil$	:	Smallest integer $\geq$ m.
$\binom{m}{r}$	:	Combinatorial function $^{m}C_{r}$.
$\square$	:	End of proof.
(a,b)	:	Greatest common divisor of a and b.
f	:	1–D or 2–D vector.
f_i	:	Components of a 1–D vector f.
$v_{i,j}$	:	$(i,j)^{th}$ component of a 2–D vector v.
WHT	:	Walsh–Hadamard Transform.
IWHT	:	Inverse Walsh–Hadamard Transform.
DFT	:	Discrete Fourier Transform
IDFT	:	Inverse Discrete Fourier Transform
RM	:	Reed–Muller code.
$R(r,2^m)$	:	r^{th} order Reed–Muller code of length 2^m.
TSKN algorithm	:	Decoding algorithm for $R(r,2^m)$ due to Tokiwa et.al [46]
$2^{m-4}*R(2,2^4)$	:	2^{m-4} times simple iterated $R(2,2^4)$ code.
BCH code	:	Bose–Chaudhury–Hocquengham Code
LFSR	:	Linear feed–back shift register
B–M algorithm	:	Berlekamp–Massey algorithm for LFSR synthesis
MLFSR	:	LFSR generating prescribed multiple sequences
FT	:	Feng and Tzeng algorithms for MLFSR synthesis
FT–1	:	Multi–sequence LFSR synthesis problem based on a generalization of Euclidean algorithm [52]
FT–2	:	Multi–sequence LFSR synthesis algorithm based on a generalization of B–M algorithm [53]
H	:	Hadamard matrix.
$f \longleftrightarrow F$	:	Transform pairs f and F.
$\odot$	:	Kronecker matrix product.
f(X)	:	A binary valued function.
$\hat{f}(X)$	:	Mapping of f(X) to the set $\{1, -1\}$ of reals.
$f \otimes g$	:	Dyadic convolution of f and g.

F.G	:	Pointwise product of F and G.
$i \oplus j$	:	Pointwise EX_OR addition of bits of i and j.
f * g	:	Cyclic convolution of f and g.
r	:	Received vector.
e	:	Error vector.
c	:	Code vector.
p^∞	:	A semi—infinite sequence $p_0, p_1, p_2, \ldots$
$LC(p^N)$	:	Linear complexity of sequence p^N
$W(B^N)$	:	Hamming weight of the vector sequence B^N of length N.
Λ	:	Error locator vector.
λ	:	Auxiliary vector.
I_e	:	The ideal of connection polynomials in 2—D deconvolution approach .
I_c	:	The ideal of connection polynomials in 1—D deconvolution approach .
x_i	:	Polynomial over Boolean field.
$x_i.x_j$	:	Boolean polynomial product function of x_i and x_j.
v_0	:	First order zeroth Reed—Muller basis vector.
v_i	:	First order i^{th} Reed—Muller basis vector.
$v_i.v_j$	:	Pointwise product of RM basis vectors v_i and v_j.
V_i	:	WHT of the real vector corresponding to RM basis vector v_i.
$v_{i,j}$	:	$v_i.v_j$
V_{st}	:	WHT of real vector corresponding to pointwise product of RM basis vectors v_s and v_t.
SI code	:	Simple iterated code.
SEC—DED code	:	Single—error correcting and double—error detecting code.
P(T)	:	Permutation matrix corresponding to transformation T
$|a|b|$	:	Concatenation code vectors a and b.
R_a	:	Vector containing absolute values of components of R.

Δ	:	NAND gate delay.
$T(r,2^m)$	:	Decoding delay for r^{th} order RM code.
$T_{maj}(r,2^m)$	:	Decoding delay for r^{th} order RM code for majority logic decoding algorithms
BA–1	:	Blahut decoding algorithm for random error–correction in 2–D BCH codes
BA–2	:	Blahut's decoding algorithm for burst–error correction in 2–D BCH codes
IBA–1	:	Improved BA–1 decoding algorithm
IBA–2	:	Improved BA–2 decoding algorithm
t–REC 2–D BCH	:	A class of t–random error–correcting 2–D BCH codes
Y_k	:	β^{j_k}
X_k	:	α^{i_k}
rank(A)	:	Rank of A
$\Lambda_m(z)$	:	The common row connection polynomial satisfied by the first m rows
λ_m	:	The common auxiliary column vector satisfied by first m rows.
MSTD Algorithm	:	Mixed spectral and time domain decoding algorithm for 2–D BCH codes
S_M	:	Matrix formed from syndromes.
R	:	2–D DFT of the received vector r.
R'	:	1–D DFT of either rows or columns of r.
$\underline{R}_i'^{(r)}$	:	i^{th} row of 2–D vector R'.
$\underline{R}_j'^{(c)}$	:	j^{th} column of 2–D vector R'
S	:	1–D DFT of either columns or rows of first stage decoded r.
(o,o)	:	Column–wise configuration of a 2–D error pattern having one error in two columns.
d_{min}	:	Minimum distance of a linear code.

e	:	Error—correcting capability of a linear code $= \lfloor (d_{min}-1)/2 \rfloor$.
e_{BCH}	:	Correcting capability of BCH decoding algorithm.
e_{alg}	:	Correcting capability of decoding algorithm of [10].
e_{2-D}	:	Correcting capability of 2–D spectral decoding algorithm.
$\Lambda(z)$	:	The row connection polynomial.
$\deg(\Lambda(z))$	:	Degree of the connection polynomial.
$\phi(z)$	:	The column connection polynomial.
w_j	:	The correction number assigned to j^{th} row or column decoding.
$\Lambda_j(z)$	:	The connection polynomial of row—j
$\phi_{uc}(z)$	:	The connection polynomial formed by roots of locators of least reliable decoded rows
$I \ \square \ O$	:	Concatenated code with outer code O and inner code I.

Chapter 1

INTRODUCTION

1.1 Scope and motivation

The main thrust of the thesis is to expand the range of codes decodable using spectral techniques. Two classes of spectral transforms chosen for study are discrete Fourier transform (DFT) and Walsh–Hadamard transform (WHT), both important special cases of generalized Walsh–Hadamard transforms. Using these classes of transforms, decoding problems in the spectral domain are shown to be equivalent to appropriate deconvolution problems. This viewpoint provides a common operating principle for various decoding algorithms based on DFT and WHT, although the underlying procedures for performing deconvolution may be different. Structural properties of the transforms and codes are used to obtain simple decoding (deconvolution) algorithms. WHT techniques are applied for decoding of Reed–Muller (RM) codes of order 2 and 3, whereas DFT techniques are applied for decoding of different classes of cyclic codes.

Decoder is an important constituent of an error–correction coding system. It retrieves the transmitted information symbols from the received signal contaminated by channel noise. In order to have a good decoder, design and implementation of good decoding algorithms has attracted lot of attention in the literature. Since decoding speed puts a restriction on the overall throughput and a simpler decoder architecture reduces the cost of decoding, smallest possible decoding delay and simplest possible decoder architecture are the two important objectives of a good decoding algorithm.

Achieving these objectives of reduced delay and implementational simplicity depends primarily on the mathematical structure of the code [1]. In some cases, the structural symmetries directly suggest a simple implementation scheme. Threshold decoding scheme for the class of RM codes and Meggit decoder scheme for the class of cyclic codes may be seen as examples in this category [1]. In other cases, it may be necessary to seek alternative representations of codes for obtaining good decoding algorithms. Appropriate spectral (transform) domain representations of codes provide good means for developing efficient decoding algorithms. Resulting algorithms are referred to as spectral or transform domain decoding algorithms and they constitute an important subset of algebraic decoding algorithms. Examples of existing transform domain decoding algorithms are DFT domain decoding algorithms for BCH codes [2]–[4] and WHT domain decoding algorithm for first order RM codes [5]–[7].

The BCH decoding algorithm has remained to date the most important algorithm among spectral decoding algorithms based on DFT domain techniques. It is found that the BCH decoding algorithm can be easily modified for correcting both errors and erasures and also for decoding beyond BCH bound [2]–[4].

The concept of BCH decoding has been applied for decoding of other related classes of codes such as alternant codes and Goppa codes [8]-[9]. BCH decoding methodology has been extended for developing decoding algorithms for the class of 2-D BCH codes which are obtained by generalizing the concept of BCH codes to two dimensions [4]. Recently, some attempts have been made to decode some non-BCH cyclic codes using spectral methods [10]. A careful examination of different DFT domain decoding algorithms reveals certain basic principles which can be used to derive all these decoding algorithms.

The range of application of WHT techniques is limited to the class of RM codes which are closed under dyadic shifts [11]. Presently decoding algorithms are available only for first order RM codes. Attractiveness of WHT is that the transform computations involve integer arithmetic and WHT domain decoding operations involve mainly comparisions and substitutions. Taking clue from the DFT domain approach, it is worthwhile to look for common principles from which one can also derive WHT domain spectral algorithms.

We view some of the important existing decoding algorithms from deconvolution viewpoint and obtain, in some cases, improvement in decoding speed. We also apply the deconvolution viewpoint to obtain new decoding algorithms.

1.2 A historical perspective of spectral decoding

Historically, the first decoding algorithm to make use of spectral techniques is the WHT based decoding algorithm for first order RM codes [5]. However, the idea did not lead to further applications. The DFT domain decoding methods have their origin in the development of BCH decoding algorithm. In order to put forward our study of DFT based spectral decoding algorithms in proper perspective we now review the earlier history of coding theory upto the development of BCH decoding with emphasis on evolution of decoding algorithms and also consider theoretical developments leading to notion of spectral decoding. Post-BCH decoding algorithm developments shall be reviewed later when we discuss spectral methodologies in Chapter 2. Along with the improvements in decoding algorithms efforts have been on to develop better decoding architectures to implement various decoding algorithms. We also give a review of decoding architectures for implementing spectral decoding algorithms.

Decoding has been one of the important topics of coding theory since early times along with other topics such as code design, distance bounds, and weight structure [12]. For block codes there are different classifications of decoding methods depending upon the reliability of demodulator decision rule, decoder metric and structural properties of codes [13]. As explained earlier, decoding algorithms presented in this thesis can be classified as spectral decoding algorithms, an important sub class of algebraic decoding algorithms. A spectral domain decoding algorithm processes an appropriate transform domain version of hard decision quantized received vector. It is an incomplete decoder in the sense that given a received

vector, it either performs correction if errors are within some upper limit or decides that the errors are uncorrectable.

We shall now consider developments in the initial phase of coding theory leading to spectral decoding algorithm. Shannon, in his pioneering theorem on coding problem in the presence of noise, proved the existence of good codes having low probability of error and non–vanishing rate [14]. Shannon's proof, which as Abramson puts it "little more than existence proof but little less than constructive proof" [15], did not address the methods and problems of encoding and decoding. Practical constraints of implementation, therefore, forced researchers to concentrate on studying structured codes [1], that is, to specify a class of codes which are shown to have requisite error–correcting capability and to make use of their mathematical structure to devise simple and implementable encoding and decoding algorithms. Hamming's work led to systematic construction of single error–correcting codes along with a simple decoding algorithm [16]. Golay gave another important class of codes during the same period [17]. Muller gave the construction of, what are now known as Reed–Muller (RM) codes, from the point of view of switching theory [18] and Reed found a practical threshold decoding algorithm for correcting upto their full capability [19]. The next development was unified mathematical theory of linear codes by Slepian [20]. Decoding scheme for implementing maximum likelihood criterion for linear codes was shown to be described by standard array decoding method, and best decoder architecture for implementing standard array decoding simply consists of a syndrome computer and a look–up table. A disadvantageous aspect is that the space complexity of standard array decoding algorithm is very high even for moderate length codes and therefore search for further specialized codes continued. Among the sub–classes of linear codes the class of cyclic codes introduced by Prange is most impressive [21]. The mathematical structure for studying cyclic codes specializes to polynomial algebra from that of vector space for linear codes. Soon it became clear that encoding and decoding can be easily implemented by means of linear feedback shift registers [22], [23]. Meggit proposed a decoding scheme for cyclic codes based on cyclic property [24]. The cyclic structure simplified the decoding architecture and the associated circuitry. However, the complexity of the logic circuit still increases with the increase in the number of errors to be corrected.

Towards late 1950's, finite field algebraic approach to coding theory gained more importance. Reed and Solomon came up with idea of what are now known as R–S codes and a decoding algorithm for them that can be considered spectral in essence by hindsight [25]. R–S codes were generalized to the class of BCH codes discovered independently by Bose, Chaudhury and Hocquenghem [26]–[27]. An important feature of R–S and BCH codes is that these codes could be designed to have specified error–correcting capability. Bose and Chaudhury proved a minimum distance bound on BCH codes which contained germinating ideas for the forthcoming BCH decoding algorithm. It was Peterson who introduced the concept of "error–locator" polynomials whose roots contain locations of errors and proposed a decoding algorithm for BCH codes [23]. This was extended to non–binary case by Zierler and Gorenstein [28]. We

also like to note contribution of Mattson and Solomon in explaining the idea of BCH codes from the stand point of recurrence theory that contains important ideas of spectral domain decoding, and it has played a very useful role in theoretical developments [29]. Chien and Forney further improved the Peterson–Gorenstein–Zierler decoding algorithm for BCH codes [30]–[31]. However, the bottleneck of solving a system of equations for getting the coefficients of the error–locator polynomial continued to persist. Berlekamp proposed his famous algorithm for efficiently solving the "key equation" [32]. Massey gave an alternate simple and elegant proof from the viewpoint of synthesis of linear feedback shift registers [33].

So far we have reviewed developments of decoding algorithms leading to BCH decoding algorithm. Around the same time there were developments in the theory of discrete Fourier transform with the publication of Cooley–Tukey algorithm [34]. Application of DFT ideas in various related disciplines also received attention. Pollard discussed the concept of Fourier transforms over finite fields [35] and soon this concept was made use of in coding theory by Gore, Chien, Choy, and others [36]–[38]. However, it was Blahut who unified all these approaches and proposed a spectral viewpoint for studying coding theory [3]–[4]. A significant gain of this approach is that it reveals close connections between areas of coding and digital signal processing.

We shall now briefly review some decoder architectures for implementing spectral decoding algorithms based on DFT. Decoding architectures based on the solution of B–M algorithm have received considerable attention. However, the best decoding architecture to be used appears to be yet unknown. There are different implementations depending upon the cost and throughput considerations. Clark and Cain have reviewed various decoder architectures in [13]. A highly parallel realization based on Massey's idea [33] for implementing B–M algorithm has been used to achieve data rates of 10–40 Mbs/sec [39].

The next phase of developments came with advances in VLSI technology. Berlekamp bit serial algorithm [40] has been instrumental in VLSI design and implementation of encoders for the class of R–S codes discussed in [41]–[42]. Berlekamp subsequently in his paper on "Error correcting technology" discussed the economics of implementing decoder and encoder in VLSI technology [43]. He also discussed cost/performance trade–off for different BCH decoding architectures. Reed et.al. have proposed systolic architectures for decoding of R–S codes based on spectral domain decoding for error correction in [44]–[45].

1.3 Summary of results

1.3.1 WHT domain decoding algorithms

A mapping from binary {0,1} to real {−1,1} is necessary in order to apply WHT techniques. Under this mapping, the EX–OR addition in binary field is mapped to pointwise multiplication in the real field and therefore the received signal in the time domain can be considered the pointwise product of channel error and the transmitted codeword. In the transform domain the received spectrum can therefore be

considered as dyadic convolution of the error spectrum and the transmitted codeword spectrum. The error spectrum can accordingly be visualized as the impulse response of a linear filter which convolutionally distorts the transmitted codeword spectrum. The decoding problem can then be formulated as deconvolution of the WHT of received signal to get back the codeword spectrum given some knowledge about error. It is found that WHT spectrum of RM codes can be grouped into several histogram classes, where a histogram class consists of all transformed codewords having the same distribution of spectral values (for histogram only absolute values are considered). It is shown that maximum possible deviation of the WHT spectrum of the received signal from the WHT spectrum of the transmitted codeword is linearly related to number of errors occurred. It turns out that the decoding process requires only simple operations such as comparison and substitution.

Let $R(r, 2^m)$ denote r^{th} order RM code of length 2^m. The WHT spectrum of $R(1, 2^m)$ is known to have only one nonzero component having a value either 2^m or -2^m. In other words, $R(1, 2^m)$ has a single histogram class consisting of a single component with the absolute value 2^m and the remaining 2^m-1 zero valued components. Our contribution is in extending this type of WHT characterization to second order RM codes. We show that the WHT spectrum of $R(2, 2^m)$ has $\lfloor m/2+1 \rfloor$ different classes, j^{th} class containing 2^{2j} nonzero spectral locations having value $\pm 2^{m-j}$ for $j = 0$ to $\lfloor m/2 \rfloor$. The proof of results are based on automorphism group property of RM codes and a theorem concerning canonical representation of multivariable polynomials over a binary field. Unfortunately histogram characterization could not not be extended to third and higher order RM codes, primarily because of non—availability of appropriate canonical forms.

Our next contribution is in using this feature of WHT spectrum of RM codes for their decoding. To proceed with the decoding, however, it is necessary to know how WHT histogram of the code is modified due to channel errors. This computation is in general quite tedious except in case of small errors. Fortunately, it is found that knowledge of range of variation of spectral values is helpful for decoding purpose. The range of variation of a spectral value is linearly related to number of errors occurred.

The possibility of recovering the histogram of the transmitted codeword from that of the received word is based on the following two conditions: first, the sign of a nonzero spectrum should not change due to channel error and second, different spectral values within a histogram class are far enough so that their ranges of variation are distinct. An analysis of these conditions shows that it is not possible to decode $R(2, 2^m)$ in a straightforward manner, the exception being $R(2, 2^4)$, which is first nontrivial single—error correcting second order RM code of length 16. We give a decoding algorithm for single—error correcting $R(2, 2^4)$ code and utilize it to develop a decoding algorithm for general $R(2, 2^m)$.

In order to obtain WHT domain decoding algorithm for general $R(2, 2^m)$, we utilize superimposition property of a RM code to decompose a larger RM code into smaller RM codes having efficient decoding algorithms, which can be combined to provide a decoding algorithm correcting upto full error correcting

capability of the larger code. In [46] Tokiwa et.al have given an algorithm (which we refer to as TSKN algorithm) based on superimposition property which reduces the larger RM decoding into several smaller single–error–correcting and double–error–detecting RM codes having an efficient decoding algorithm. They have also shown that TSKN algorithm has better delay compared to conventional majority logic decoding algorithms for RM codes.

We propose an alternative way of decomposition that reduces decoding of $R(2,2^m)$ to (m–4) fold decoding of $R(1,2^{m-1})$ and a single decoding of $R(2,2^4)$ repeated 2^{n-4} times. (For brevity the repeated code is represented as $2^{m-4}*R(2,2^4)$). A new WHT domain decoding algorithm for decoding of $2^{m-4}*R(2,2^4)$ is developed on the lines of WHT domain decoding algorithm for $R(2,2^4)$. This new WHT domain algorithm and the well–known WHT domain decoding algorithm for $R(1,2^m)$ are combined to provide decoding algorithm for $R(2,2^m)$. We propose two variations of the decoding algorithm for $R(2,2^m)$, one of which avoids the need for inverse WHT and also simplifies the process of recovering information bits.

We show in a similar way that decoding of third order RM codes reduces to decoding of (m–5) fold $R(2,2^{m-1})$ and a single decoding of $2^{m-5}*R(3,2^5)$. WHT domain decoding algorithms for $R(3,2^5)$ and $2^{m-5}*R(3,2^5)$ are given and utilized for developing a decoding algorithm for $R(3,2^m)$ by combining decoding algorithms for $2^{m-5}*R(3,2^5)$ and $R(3,2^5)$.

There are important differences in the workings of the TSKN decoding algorithm and the WHT domain algorithm, apart from different ways of decomposing a RM code. Both the methods at some stage have to do decoding of repeated codes. TSKN method attempts to decode each code block of the repeated code and then ascertains the validity of decoding. If the result is not valid, it takes the next code block and repeats the process. The number of attempts varies from one to as many as the number of repeated code blocks. WHT domain decoding algorithms attempts to decode the entire repeated code in a single step. In case there are ambiguities regarding the spectral pattern of the code, it requires one more step to complete the decoding. Therefore, the number of attempts is at most 2. This feature may simplify the decoder architecture compared to TSKN method. We give a decoder architecture based on the WHT domain decoding algorithms for second and third order RM codes. An estimate of the decoding delay with this architecture is given. The delay is shown to be significantly less compared to TSKN algorithm, which in turn has smaller delay compared to other decoding algorithms for higher order RM codes.

1.3.2 DFT domain decoding algorithms

Both 1–D and 2–D DFT domain algorithms for decoding cyclic codes depend upon characterization of cyclic codes as having a specified set of spectral components identically equal to zero. As a consequence the DFT spectrum of the received vector at these locations directly gives the value of the corresponding error spectrum. The direct availability of partial information on error is in contrast to its non–availability

in case of WHT. By introducing an auxiliary vector which is zero at locations corresponding to error locations and considering their pointwise product and transforming the pointwise product vector, a zero–output convolution involving the error spectrum is formed. DFT domain decoding can therefore be formulated as 'zero–output deconvolution' with the partial knowledge of error spectrum. (In coding theory practice, the error spectrum is alternatively known as syndrome.) The other convolving component, the DFT of the auxiliary vector, is also of importance in that its zeros are related to the locations of the error; the associated polynomial is known as the error–locator polynomial. In the DFT domain the deconvolution takes the form of problem of finding the connection polynomial of least order LFSR generating the entire error spectrum with the known error spectrum at the zero locations of the code as the initial values of the LFSR. In view of this the error–locator polynomial is also referred to as the connection polynomial.

We treat 1–D and 2–D decoding algorithms in a single framework. Analysis of a typical DFT decoding algorithm reveals three important stages: the error spectrum computation stage, the error location stage and the error evaluation stage. Any DFT domain decoding approach centers around the error location stage which involves formulation and computation of error location polynomials. Any particular decoding approach is typified by the nature of the deconvolution corresponding to the nature of the convolution of the transform domain vectors of the corresponding error locator and error spectrum polynomials. For efficient computation, the error location stage requires some structure on the zero locations of the code; this also influences the error evaluation stage. Error evaluation stage can be completed in many ways and an appropriate choice is made so as to reduce the overall decoding complexity.

We study two distinct DFT domain decoding approaches: 2–D deconvolution approach based on Sakata's algorithm for 2–D LFSR synthesis [47]–[49] and 1–D deconvolution approach based on a 2–D linear complexity theorem. The latter approach is our contribution based on a 2–D extension of Blahut's 1–D linear complexity theorem [50].

In the 1–D BCH decoding the concept of an error locator is to associate a position 'i' with α^i, where α is the primitive n–th root of unity (n is the length of the code). The basic idea of generalization from the standard 1–D BCH case is the association of a nonzero error position (i,j) with a 2–D error–locator (α^i, β^j) corresponding to row and column indices of the error position. Then the 2–D convolution of the transform domain vectors of the corresponding error–locator and error spectrum polynomials is zero and therefore the decoding problem is formulated as a 2–D DFT zero–output deconvolution. The error–locator polynomials or recurrence polynomials can be interpreted as two variable polynomials. This interpretation immediately suggests that all polynomials that satisfy a given array of 2–D error spectrum constitute an ideal. Then the problem of LFSR synthesis is shown to be equivalent to finding a "distinguished basis" of the ideal which is minimal in the sense of some total ordering for a 2–D array. Sakata has given an algorithm for efficiently computing such a basis which is equivalent to finding an algorithm for 2–D LFSR

synthesis [47]–[49]. It is to be noted that both the syndrome and LFSR are 2–D in nature. The error–evaluation stage consists of either solving for the roots of the system of connection polynomials for individual error positions in the binary case or performing 2–D recursive extension of the known error spectrum to obtain the complete error spectrum.

We propose an alternative approach to obtain DFT domain spectral decoding algorithms centered around a generalization of Blahut's 1–D complexity theorem for a linear recurring sequence. Complexity of a sequence is defined as the minimal order of LFSR generating the sequence. This notion of complexity has proved to be very useful in deriving minimum distance bounds and in determining the weight structure of codes. We show that it can also play a crucial role in the formulation and design of decoding algorithms. This fact is more evident in the 2–D decoding algorithms.

Blahut's 1–D complexity theorem states that the minimal order of LFSR generating a time (or frequency) domain sequence is equal to the Hamming weight of transformed sequence in frequency (time) domain [50]. Roots of the connection polynomial correspond to locators of the nonzero components of the time domain vector. We show that the notion of linear complexity is easily generalized to a 2–D sequence as follows. The minimal order of common LFSR generating all row (column) subsequences is equal to number of nonzero columns (rows) of 2–D transformed sequence. Roots of the common row (column) connection polynomial give the locators of columns (rows) corrupted by the error.

The basic idea of our approach is to associate a nonzero error position (i,j) with a 1–D error–locator corresponding to either row (α^i) or column (β^j) index of the error position. Then the 1–D convolution of row (column) error locator vector with each of row (column) subsequences of the 2–D error spectrum is zero. A typical 2–D decoding problem is then viewed as equivalent to solving several 1–D deconvolution problems which for the DFT case are equivalent to synthesizing several 1–D LFSRs corresponding to the row or column subsequences. Note that in contrast to 2–D deconvolution approach, LFSR synthesis is treated as 1–D whereas the error spectrum is 2–D in nature. In the error evaluation stage the recursive extension has to be done in both the column and row directions. It is also easy to see that the 2–D span of locations corresponding to product of zeros of the common row and column connection polynomials contains all the error positions. This makes 1–D deconvolution approach more elegant to take care of 2–D burst–errors compared to 2–D deconvolution approach. As the 1–D deconvolution approach is based on 1–D LFSR synthesis, this may imply milder constraints on the structure of zero locations, for a given error–correcting capability, compared to 2–D deconvolutional approach which is based on Sakata's algorithm for 2–D LFSR synthesis. A decoding example is given to illustrate this advantage of 1–D deconvolutional approach over 2–D deconvolutional approach.

For a better understanding of various DFT decoding algorithms, the relationship between error–locator polynomials that arise in 2–D and 1–D deconvolution methods is of interest. A look at the ideal structure of error–locator polynomials reveals that the ideal formed in the latter case is a sub–ideal

of ideal formed in the former case. This is because in the former approach we are interested in individual error locations whereas in the latter approach we are interested in the span of error locations. Therefore in the 2–D deconvolution method the error–locator polynomials can be algebraically solved for individual error locations whereas in the 1–D deconvolution method solution of the equations do not give exact locations of errors but only span of error locations. An algorithm due to Fitzpatrick et. al. in [51] is invoked to obtain the set of error–locator polynomials in 2–D deconvolutional approach using the error–locator polynomials obtained in the 1–D deconvolutional approach. We note that this indirect way to computing the error–locator polynomials is not efficient from the decoding viewpoint. However, this establishes the theoretical connection between these two approaches and one can continue in either approach to obtain valuable insights. As 1–D deconvolution approach is flexible to handle both random and burst errors, we have preferred to use 1–D deconvolution approach in order to obtain decoding algorithms for various classes of cyclic codes.

a) DFT domain decoding algorithms for 2–D BCH codes

We begin with applications of 1–D deconvolutional approach to look into decoding algorithms for the class of 2–D BCH codes which are defined as having a contiguous array of 2t x 2t zeros. These codes have simple structure that helps in understanding the basic principles of 2–D decoding. Blahut has given a decoding algorithm for random errors correcting upto minimum distance bound of 2t+1 [3]–[4]. He has also given a decoding algorithm for correction of some patterns of 2–D burst errors.

We take a different approach to derive Blahut's algorithm. This approach provides a better insight into the working of the decoding algorithm. The inter–relationship between polynomials are viewed from the point of view of the linear recurrence theory. This shows that certain computations can be cut down compared to Blahut's decoding method. In particular, it is shown that at most "t" passes through B–M algorithm are required to find the connection polynomial as opposed to "2t" or more in Blahut's case. An "Improved Blahut's algorithm" for random error–correction incorporating the above mentioned modifications is presented. Improvements in algorithms can also be used to advantage for decoding for burst–error correction. Another significant question in the context of 2–D BCH codes is that of maximum correction capability of the structure of zero locations in the form of 2t x 2t contiguous array. We employ 2–D complexity theorem to establish an error–correcting limit that the structure of zero locations allows. As a result of these investigations it follows that Blahut's algorithm is best possible in that it corrects upto the limit. Next we introduce a class of Γ 2–D BCH codes that has less number of zeros compared to the class of 2–D BCH codes but has the same error–correcting capabilities. We employ 2–D complexity theorem to show that the error–correcting limit cannot be further improved. As it is not possible to apply Blahut's decoding algorithm directly, we give a decoding algorithm to correct upto the limit as shown by complexity theorem arguments. A mixed spectral and time domain implementation of Blahut's original as

well as modified decoding algorithms is discussed. A comparative study of complexities of various implementation schemes for decoding of 2–D BCH codes is undertaken. We also discuss an alternative approach to decode the class of 2–D BCH codes on the basis of multiple LFSR synthesis algorithm proposed by Tzeng and Feng [52]–[53]. It is shown that multiple LFSR based decoding algorithms for 2–D BCH codes can correct more pattern of burst errors.

Next we consider a class of 2–D BCH codes from the perspective of only random error correction and give a decoding algorithm for these codes. This study intuitively helps in evaluating the requirements on the structure of zero locations to correct different types of random and burst errors.

b) DFT domain decoding algorithms for cyclic codes

We next apply the 1–D deconvolutional approach to give decoding algorithms for non–BCH cyclic codes. These are not as structured as BCH codes. Nevertheless their performance is comparable to BCH codes. Though some attempts to develop decoding algorithms for such codes have been made, so far there are no decoding algorithms that apply to a larger class of codes [3], [10]. In our approach too, it is not simple to identify and characterize a general class of codes having similar decoding algorithm. However, we hope that the given 1–D deconvolutional approach provides basic principles which can be tailored for decoding of specific codes.

The main idea in developing the decoding algorithms is to analyze the changes in the degree of connection polynomials, due to different configurations of error patterns of a given weight, upto some limit which should desirably coincide with the correcting limit $[d_{min} -1/2]$ of the code. The decoding algorithms proceed in stages anticipating various possible error configurations, finding appropriate connection polynomials, and testing for their validity. Validity tests includes periodicity tests on error spectrum, testing for conjugacy constraints and checking for consistency of row (column) connection polynomials with known error spectrum along that row (column). These steps naturally lead to a relatively more complex decoding algorithm.

The main effort is how to find the appropriate row or column connection polynomial for a given configuration of errors using all possible information about the error spectrum. This is motivated by the engineering considerations and helps in a better understanding of the algorithm. The approach may be seen to carefully balance the requirements of algebraic complexity and implementation complexity.

Some of the novel features of the 2–D spectral decoding algorithms for non–BCH cyclic codes are:
(1) The algorithms sometime allow interesting variations of Chien search and trial–and–error methods.
(2) The algorithms can be modified for erasure correction on a limited scale on the lines of standard BCH and Feng–Tzeng multi–sequence shift–register synthesis algorithms [54].
(3) The algorithms are particularly suited for correction of periodic erasure bursts.

We have given 2–D decoding algorithms for both 1–D cyclic and 2–D cyclic codes. The decoding algorithms for 1–D cyclic codes are based on the 2–D DFT characterization of 1–D cyclic codes. We have chosen several 1–D cyclic codes of lengths 21, 33, 35, 45, 51 from those complied in [55]. Decoding algorithms for these codes exhibit different techniques and all of them decode upto the error–correcting capability of codes. For some of these codes an algebraic decoding algorithm has been proposed for the first time. We have also identified a class of t–random error–correcting codes of area 3 x n and 5 x n, with 2t consecutive zeros in row–0 and t consecutive zeros in any other row, having similar decoding algorithm.

Some good 2–D concatenated cyclic codes have been compiled by Jorn Jensen [56]. Decoding methods for these codes make use of concatenated property of codes. We consider almost all good codes of length 45,49 and 75 given in [56] and give decoding algorithms, some of them being good ones, based on our approach, without needing details of the components of concatenated codes.

Achievements and shortcomings of the present thesis have to viewed in the light of an important result concerning the complexity of hard–decision decoding. Berlekamp et.al. have shown that hard decision decoding problem posed in time domain is non–polynomial deterministic in the sense of complexity theory [57]. This strongly implies that given an arbitrary code it is very rare to obtain a polynomial time algorithm for retrieving the codeword from the received word. There are cases when a code is endowed with structure so that, after suitable transformation or otherwise, the decoding problem can be formulated to be solvable in polynomial time. The class of BCH codes and first order RM codes are examples of codes having tractable polynomial time decoding algorithm. Therefore, our efforts may be seen as an attempt to enlarge the scope of codes having tractable decoding algorithms.

We now summarize the results of the thesis. Our starting point is that decoding problems in the WHT and DFT domain are equivalent to appropriate deconvolution problems. We have used relevant properties of transforms along with relevant properties of codes to obtain good decoding algorithms. It is to be noted that WHT domain method gives directly the codeword spectrum whereas DFT domain method gives error estimates directly. Another distinctive feature of WHT approach is to use histogram properties of WHT spectral coefficients to separate features of error and codeword, whereas in the DFT domain approach the separation is obtained based on the characterization of a cyclic code having specified DFT components identically zero. We next formulated methodologies for decoding and applied them to obtain decoding algorithm for RM codes of order 2 and 3 and various classes of cyclic codes. These results illustrate the importance of spectral methods to obtain tractable decoding algorithms. The spectral decoding algorithms also lead to decoder implementation schemes having reduced delay and simpler decoding architectures.

1.4 Outline of chapters

Chapter 2 is devoted to studying basic principles of WHT and DFT spectral decoding and to formulate various decoding methodologies. Section 2.1 gives definitions and basic properties of WHT and DFT. In Section 2.2 it is shown that decoding problems in both the DFT and WHT domains can be interpreted as appropriate deconvolution problems. Section 2.3 presents decoding approaches for RM codes based on WHT domain techniques. Well-known WHT domain decoding algorithm for first order RM codes is reviewed and used as a model for obtaining decoding approaches for higher order RM codes. Section 2.4 begins with a review of various existing DFT domain decoding algorithms for cyclic codes. A closer examination of these algorithms in the deconvolutional formulation leads to two DFT domain decoding approaches, namely, 1-D deconvolution approach based on 2-D linear complexity theorem and 2-D deconvolution approach based on Sakata's algorithm [47]-[49] for 2-D LFSR synthesis. Section 2.5 gives a discussion on complexity theorem highlighting its role in transform domain decoding. This is followed by a presentation of 1-D deconvolution approach to DFT domain decoding algorithms. Section 2.6 considers 2-D deconvolution approach based on Sakata's 2-D LFSR synthesis algorithm. A discussion on the inter-relationship of connection polynomials obtained in these two approaches is given. Reasons for preferring 1-D deconvolution approach for further investigations are explained.

In Chapter 3 we consider a novel way of decoding $R(2,2^m)$ and $R(3,2^m)$ using WHT domain techniques. Section 3.1 reviews some of the important properties of RM codes. Section 3.2 gives a review of TSKN decoding algorithm for RM codes and explains motivations for introducing WHT domain decoding algorithms for RM codes. In Section 3.3, it is proved that WHT spectrum of $R(2,2^m)$ consists of $\lfloor m/2 \rfloor + 1$ different histogram classes. Some of the difficulties in extending histogram characterization to higher order RM codes are discussed. Section 3.4 considers application of histogram characterization results to WHT domain decoding. Methods for analyzing changes in the histogram of WHT spectrum of RM codes due to channel errors are discussed and conditions for recovering histogram patterns of transmitted codewords are elucidated. Analysis of these conditions for $R(2,2^m)$ shows that $R(2,2^4)$ is the only nontrivial case for which WHT domain techniques for full error correction can be applied directly. Accordingly, WHT decoding algorithm for $R(2,2^4)$ is given and used for WHT domain decoding of $2^{m-4}*R(2,2^4)$. Section 3.5 begins with a decomposition algorithm of $R(2,2^m)$ in terms of subcodes for which WHT domain decoding techniques can be applied. A WHT based algorithm for $R(2,2^m)$ is given in terms of WHT domain decoding of $R(1,2^{m-1})$ and WHT domain decoding of $2^{m-4}*R(2,2^4)$. An improved version of WHT domain decoding algorithm for $R(2,2^m)$ which avoids inverse computation of WHT and also simplifies retrieving information bits is also given. Section 3.6 extends these techniques to obtain WHT domain decoding algorithm for $R(3,2^m)$. In Section 3.7, decoder architectures for WHT domain decoding of $R(2,2^m)$ and $R(3,2^m)$ are given. Based on this decoding architecture an estimate of decoding delay is obtained and it is shown that for $m \geq 8$, WHT domain decoding algorithms have less delay compared to TSKN approach.

Chapter 4 is concerned with the application of 1–D deconvolution approach for decoding of 2–D BCH codes. Section 4.1 reviews the nature of 2–D error bursts so far reported in the literature. In Section 4.2, Blahut's decoding algorithm for random error correction in 2–D BCH codes is reviewed and a new exposition of the same is given in the light of 1–D deconvolution approach. Certain modifications are suggested for improving the speed of the decoding algorithm and an improved decoding algorithm is presented. Section 4.3 presents an exposition of Blahut's algorithm for burst–error correction in 2–D BCH codes and some improvements in the burst error correcting algorithm are suggested. Error–correcting capabilities of 2–D BCH codes are investigated in Section 4.4. It is shown that Blahut's algorithm is the best possible algorithm in that it corrects upto error–correcting limit of the code. Section 4.5 introduces a new efficient class of Γ 2–D BCH codes which is shown to have the same error correction capabilities as the class of 2–D BCH codes. A decoding algorithm correcting upto the error–correcting capability of these codes is also presented. Section 4.6 discusses mixed spectral and time domain implementation of decoding algorithms for various classes of 2–D BCH codes. Section 4.7 considers an alternative approach for decoding the class of 2–D BCH codes based on Multiple LFSR synthesis algorithm due to Feng and Tzeng [52]–[53]. It is shown that this decoding approach can correct more burst error patterns compared to Blahut's decoding algorithm. Section 4.8 considers a class of t–random error correcting 2–D BCH codes along with its decoding algorithm. This study also helps in intuitively evaluating the requirements on the structure of zero locations to correct various combinations of burst and random errors.

Chapter 5 considers application of 1–D deconvolution approach to DFT domain decoding methodology to obtain 2–D decoding algorithms for 1–D cyclic codes. Section 5.1 reviews previous approaches for decoding non–BCH cyclic codes. In Section 5.2 general outlines of 2–D spectral decoding algorithms are explained and some of the novel features of these algorithms are mentioned. Classification of 2–D spectral decoding algorithms into type–1, type–2 and type–3 is discussed. Notations required for explaining and presenting various decoding algorithms are also explained. Sections 5.3 to 5.9 give 2–D decoding algorithms for 1–D cyclic codes of length 21, 33, 35, 39, 45, 51 and 63.

Chapter 6 is concerned with decoding algorithms for 2–D cyclic codes. A table of codes for which 2–D decoding algorithms are given is presented in Section 6.1. Section 6.2 considers decoding algorithms for codes of length 45, area 3 x 15. Section 6.3 gives decoding algorithms for codes of length 49, area 7 x 7. Section 7.4 is devoted to decoding algorithms for codes of length 75, area 5 x 15.

Chapter 7 summarizes important contributions of the thesis and suggests some topics for further research.

Chapter 2

DECODING APPROACHES BASED ON SPECTRAL DOMAIN DECONVOLUTION FOR REED-MULLER AND CYCLIC CODES

This chapter is devoted to studying the principles underlying the formulation and design of spectral (transform) domain decoding algorithms for RM and cyclic codes. It is shown that decoding problems for RM and cyclic codes can be viewed as appropriate deconvolution problems in WHT and DFT domain respectively. This viewpoint helps us in identifying decoding methodologies for RM and cyclic codes depending upon the nature of deconvolution in WHT and DFT domain respectively. The study of such decoding approaches not only provides an alternative way of understanding the existing spectral domain decoding algorithms, but also provides a general framework for designing new decoding algorithms in the spectral domain.

WHT and DFT transform domain decoding (deconvolution) techniques differ in the arithmetic used, the way of formulating the decoding problem and also in the process of decoding. A comparative study of decoding approaches based on these two transforms brings into focus common principles involved in decoder operations and provides insights into the nature of the underlying algebraic decoding algorithms.

Section 2.1 reviews basic properties of WHT and DFT. In Section 2.2 we show that WHT and DFT domain decoding problems can be formulated as appropriate deconvolution problems in respective transform domains. Section 2.3 is devoted to WHT domain decoding approaches to RM codes. The well-known WHT based decoding algorithm for first order RM codes is re-examined and used as a model for formulating the decoding problem for higher order RM codes. In Section 2.4 we review existing DFT domain encoding and decoding algorithms. The need for a common framework to analyze both 1-D and 2-D DFT domain decoding algorithms is explained. We study two distinct DFT decoding approaches: 1-D deconvolution approach based on a 2-D linear complexity theorem, and 2-D deconvolution approach based on Sakata's algorithm for 2-D LFSR synthesis. Section 2.5 begins with a discussion on the role of complexity theorem in decoding. A generalization of complexity theorem from 1-D to 2-D is stated and then 1-D deconvolution approach to DFT domain decoding based on the complexity theorem is explained. Formulations of decoding problems for certain well-known codes are re-examined as illustrations of this approach. Section 2.6 presents 2-D deconvolution approach to DFT domain decoding based on Sakata's algorithm for 2-D LFSR synthesis. Next we study interrelations between the set of connection polynomials obtained in 2-D deconvolution approach to those obtained in 1-D deconvolution approach. Reasons for preferring the latter approach for further investigations are explained.

2.1 Review of properties of spectral transforms

It is shown in [58] that generalized Walsh Hadamard transforms provide a natural setting for the study of a wide range of linear block codes including cyclic and RM codes. Well–known finite field DFT and WHT are two important special cases of these generalized transforms. The basis vectors of DFT form eigen vectors of cyclic shift invariant linear systems whereas the basis vectors of WHT form eigen vectors of dyadic shift invariant linear systems. Some of the important properties of WHT and DFT domain representations are given below for later reference.

2.1.1 Definition and properties of WHT [11]

If $f = (f_0, f_1, ..., f_{n-1})$ denotes a vector of reals, then the WHT $F = (F_0, F_1, ..., F_{n-1})$ of f is given by

$$F_j = \sum_{i=0}^{n-1} (-1)^{\sum_{k=0}^{m} i_k j_k} f_i \; ; \; 0 \leq j \leq n-1, \; n=2^m \; .$$

The above equation can be written in the matrix form as

$$F = Hf$$

where H is Hadamard matrix whose $(i,j)^{th}$ element is given by $(-1)^{\sum_{k=0}^{m} i_k j_k}$. The matrix H can be written in the following form

$$H = H_2 \odot H_2 \odot H_2 \odot ... \odot H_2 \; (m \text{ times})$$

where H_2 is a Hadamard matrix of order 2 and $\odot$ denotes Kronecker matrix product.

The inner product of any two rows of H is

$$\sum h_{ik} h_{jk} = \begin{cases} 2^m & \text{if } i=j \\ 0 & \text{otherwise} \end{cases},$$

that is, the rows of H are orthogonal. The inverse of H is accordingly given by

$$H^{-1} = 1/2^m \; H.$$

It is clear that the computation of inverse of H is identical to the computation of H except for final division by 2^m.

Let $f(X)$ denote a binary valued function and $\hat{f}(X)$ be its corresponding mapping to the set $\{1, -1\}$ of real numbers. Let $\hat{F}$ denote the WHT of $\hat{f}$. The following properties are consequences of orthogonality of the WHT basis vectors.

(a)   The sum of all the spectral components of $\hat{F}$ is $\pm 2^m$.

(b)   The range of values for individual spectral components is $\{-2^m, -2^m+2, ..., 0, ..., 2^m-2, 2^m\}$. The maximum possible absolute value is 2^m. When one of the spectral coefficients is maximum–valued, the remaining spectral coefficients are all zero valued.

If $f \leftrightarrow F$ and $g \leftrightarrow G$ are two WHT pairs, then we have

$$f + g \longmapsto F + G$$
$$f \otimes g \longmapsto F.G$$

where $(f \otimes g)_j = \sum_{i=0}^{n-1} f_i g_{i \oplus j}$; $0 \le j \le n-1$, $n = 2^m$; '$\otimes$' represents dyadic convolution; $i \oplus j$ denotes pointwise ex-or addition of bits of i and j; F.G denotes pointwise product of vectors F and G.

In the case of real vector $\hat{f}$ derived from the binary vector f, the components take values from the set $\{1, -1\}$ and the spectral components can be given following interpretation. The jth spectral coefficient is equal to difference of the number of disagreements from the number of agreements between j^{th} row of H and the real vector $\hat{f}$. This is seen to be the same as the correlation between j^{th} row of H and the given vector $\hat{f}$. Alternatively, the jth spectral value is equal to ($2^m - 2^*$Hamming distance between j^{th} row of H and $\hat{f}$). This interpretation of WHT spectral coefficient in terms of Hamming distance is useful in understanding WHT domain decoding algorithms for RM codes.

2.1.2 Definition and properties of DFT [4]

Basic knowledge of finite fields is assumed. Let f denote a vector of length n whose components take values from GF(q). Let n divide q^m-1 for some m, and let α be an element of order n in $GF(q^m)$. Then the finite field DFT F of f is given by

$$F_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i \quad 0 \le j \le n-1$$

The inverse transform is defined by

$$f_i = 1/n \sum_{j=0}^{n-1} \alpha^{-ij} F_j \quad 0 \le i \le n-1 ,$$

where $1/n$ is an integer in the field modulo p, the characteristic of GF(q). For extension $GF(2^k)$, n is odd, $p = 2$ and therefore, $1/n$ is 1. Similarly, 2-D DFT of a 2-D array v of area $n_1 \times n_2$ over GF(q) is defined as follows. Let α and β be elements in $GF(q^m)$ of order n_1 and n_2 respectively. Then

$$V_{i,j} = \sum_{i'=0}^{n_1-1} \sum_{j'=0}^{n_2-1} \alpha^{ii'} \beta^{jj'} v_{i',j'} ,$$

where $0 \le i \le n_1-1$ and $0 \le j \le n_2-1$.

$$v_{i',j'} = 1/n_1 n_2 \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \alpha^{-ii'} \beta^{-jj'} V_{i,j} ,$$

where $0 \le i' \le n_1-1$; $0 \le j' \le n_2-1$

It is important to note that finite field DFT does not exist for all n, unlike real field DFT. For example, it is not possible to have DFT of length 8 over GF(2). We now consider some important properties of DFT.

1)   Linearity and Convolution : If f⟷F and g⟷G are two DFT pairs, then we have

$$f + g \longmapsto F + G$$

$$f * g \longmapsto F.G$$

where F.G denotes pointwise multiplication of vectors F and G, and $(f^*g)_j = \sum_{i=0}^{n-1} f_i g_{(j-i)_n}$; $0 \leq j \leq n-1$,

$(i-j)_n$ denotes $(i-j)$ evaluated modulo n. In the 2–D case the definition of convolution is given by

$$(f^*g)_{i,j} = \sum_{i'=0}^{n_1-1} \sum_{j'=0}^{n_2-1} f_{i',j'} g_{(i-i')_{n_1},(j-j')_{n_2}}$$

If we associate with vector f over GF(q), a polynomial $f(x) = \sum_{i=0}^{n-1} f_i x^i$, then the convolution of vectors f and g can be alternatively expressed as

$$(f^*g)(x) = f(x)g(x) \text{ modulo } (x^n-1)$$

For 2–D case, we associate 2–variable polynomials in the natural way and the convolution is appropriately expressed as

$$(f^*g)(x,y) = f(x,y)g(x,y) \text{ modulo } ((x^{n_1}-1), (y^{n_2}-1))$$

Since the cyclic code is interpreted as an ideal in the polynomial algebra modulo (x^n-1) [1], it can be shown that in the spectral domain a cyclic code is characterized by having specified spectral components identically equal to zero [4]. For 2–D case also, this interpretation holds true as 2–D cyclic codes are ideals in 2–variable polynomial algebra modulo $((x^{n_1}-1), (y^{n_2}-1))$ [59].

2)   Conjugacy constraints : This property is finite field counterpart of conjugate symmetry of classical Fourier transform of a real vector, i.e., $V^*(f) = V(-f)$. If the input vector v is over GF(q) and the transform vector V is over $GF(q^m)$, an extension field of GF(q), then the components of V satisfy

$$V_j^q = V_{(qj)_n} \; ; \; j = 0, 1,..., n-1.$$

Thus the set of spectral components with indices $\{j, (jq)_n, (jq^2)_n,...,(jq^{r-1})_n\}$, where r is the least integer such that $j \cong jq^r \pmod{n}$, are related by conjugacy constraints and referred to as a chord or conjugacy class. The least integer j in the conjugacy class is called the conjugacy leader, by convention.

The implication of conjugacy constraints for characterization of 1–D and 2–D cyclic codes is that if a particular spectral component V_j is zero, so are the spectral components belonging to its conjugacy class.

2.2 Deconvolution formulations of spectral domain decoding

It is known that in the time domain the decoding problem is formulated as the maximum likelihood sequence detection problem with the criterion of minimizing probability of block error. By virtue of the assumption of memoryless channel, vector sequence detection problem is further simplified to finding that

code vector which is least distant, in terms of Hamming metric, from the received vector. Similarly, it will be of interest to look at the formulation of decoding problem in the spectral domain, in general, from the standpoint of what inputs are available and what are the outputs sought. In the literature the nature of spectral domain decoding formulation is not satisfactorily clarified. We now show that decoding problem, viewed in spectral domain, for both DFT and WHT, reveals a close similarity, although decoding operations are in different fields. As an interesting offshoot of this study we discuss an alternative way of decoding in DFT domain.

The point of similarity is that both the WHT and DFT domain decoding problems can be formulated, in essence, as deconvolution problems. In the area of digital signal processing deconvolution refers, in general, to cases when given the convolved data y (= x * h), it is desired to determine input and/or the impulse response of the linear invariant system, assuming available partial knowledge of input and/or impulse response. The procedure is to use structural properties of signals to compute the unknown components. For example, in the context of seismic processing, the low pass and high pass filtering of the complex cepstrum of the received data separates the signal features belonging to x and h [60]. We shall show later that both WHT and DFT are endowed with useful properties which can be exploited to obtain suitable deconvolution algorithms.

The channel error is modeled as additive distortion to the transmitted code vector. The received vector is therefore represented as $r = c \oplus e$. In order to apply WHT techniques it becomes necessary to
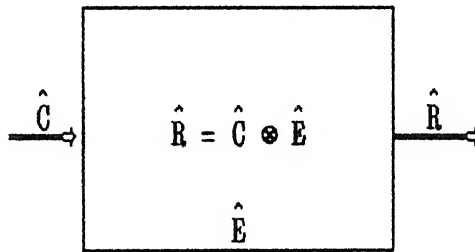
Figure 2.2.1 Convolutional representation of $\hat{R}$ in WHT domain

map binary alphabet to real {1,−1} and this maps the binary EX−OR addition to pointwise multiplication. If $\hat{r}$ represents the real vector corresponding to r, then we have $\hat{r} = \hat{c}.\hat{e}$, where '.' denotes pointwise multiplication of the vector components. Upon spectral transformation this becomes

$$\hat{R} = \hat{C} \otimes \hat{E}. \tag{2.2.1}$$

Therefore, the objective of a spectral decoding algorithm is to determine $\hat{C}$ given $\hat{R}$, which is straightforward deconvolution. One important thing to be noted is that it is not possible to obtain any partial information on the components of either $\hat{E}$ or $\hat{C}$. It is shown that the histogram characterization of

WHT spectrum of second and third order RM codes is the basis for obtaining simple decoding (deconvolution) algorithms combining the structural properties of WHT and RM codes.

In the case of DFT, the received vector $r = c \oplus e$ is transformed in the DFT domain as as $R = C + E$. Note that transform domain computations are now in Galois extension field unlike WHT where computations involve integer arithmetic. As the spectral characterization of a cyclic code is such that some spectral coefficients are identically zero, it is immediate that information on error spectrum at these locations is directly available to the decoder. This availability of partial information on error is the important distinguishing feature of DFT domain decoding algorithms. However, no simple direct method is available for obtaining the error components at nonzero locations.

In order to effect separation of error vector from the received vector, we introduce an auxiliary vector λ which complements the error vector e such that pointwise product $\lambda.e = 0$. By virtue of convolution theorem this implies

$$\Lambda * E = 0. \qquad (2.2.2)$$
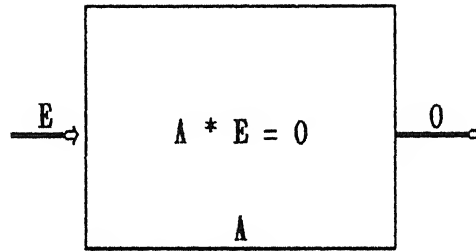
Figure 2.2.2 Linear filter connecting Λ and E in DFT domain

Equation (2.2.2) shows that the decoding formulation can be considered as "deconvolution" of convolved output 0 given that some components of E are known. For the sake of distinction from WHT case, we may refer to this formulation of DFT domain decoding as "zero output deconvolution". The deconvolution is to be performed with the constraint that the order of Λ should be as small as possible. The BCH codes are an important sub class of cyclic codes having consecutive zero locations. For these specific class of BCH codes the deconvolution problem reduces to solving a Hankel system of equations which in turn is shown to be equivalent to problem of synthesizing a linear feedback shift–register.

There is an alternate way to decode cyclic codes. By mapping the binary symbols to the set $\{1, -1\}$ of real numbers and then proceeding similar to the WHT case we have

$$\hat{R} = \hat{C} * \hat{E}. \qquad (2.2.3)$$

The difference from Equation (2.2.1) is that the convolution now is based on cyclic permutation. The decoding problem can again be formulated as deconvolution. It remains to be seen whether cyclic codes have some useful properties that can be harnessed to solve the deconvolution problem. This idea of

decoding is new and promising, and intuitively one feels that cyclic codes are highly structured to merit studies in this direction. Unfortunately, this idea struck us rather late and we had no time to explore it.

2.3 WHT domain decoding

Walsh–Hadamard processing is attractive because of integer arithmetic and unlike DFT based algorithms, no extra modules are required for arithmetic implementation. It is important to note that practically it is possible to do WHT spectral decoding on–line for low to medium data rates. As codes are defined over binary field and the domain of WHT is integer arithmetic, a suitable mapping of code bits from binary to real field is necessary before proceeding with the spectral characterization of code. We know from theory of permutation–invariant systems that the basis functions of WHT diagonalize a dyadic shift–invariant linear system. In order to get meaningful results it is necessary that the code signal space must be closed under dyadic shifts. This requirement restricts the choice of codes to the class of RM codes among linear codes.

The following are two possible mappings from binary to reals:

Mapping 1: binary 0$\longrightarrow$real 0 and binary 1$\longrightarrow$real 1

Mapping 2: binary 0$\longrightarrow$real 1 and binary 1$\longrightarrow$real -1

Table 2.3.1 gives expressions for performing logical operations in terms of real arithmetic. The notation 'c_i' denotes output and the inputs are denoted by 'a_i' and 'b_i'.

Table 2.3.1 Equivalent formulas for logical operations

Logical Operations	Mapping 1	Mapping 2
AND	$c_i = a_i.b_i$	$c_i = 1-(1-a_i)(1-b_i)/2$
EX–OR	$c_i = a_i+b_i-a_i b_i$	$c_i = a_i.b_i$

An inspection of the table reveals that any one of these mappings may be used. For, if a mapping has simple form for one of the operations, say "logical AND" operation, then the corresponding equation for the other "logical EX–OR" is somewhat involved. While computing the spectrum of a code, it is advantageous to find the spectrum of generating basis vectors of the code and then compute the spectrum of codewords generated by the linear span of basis codewords. Since for the operation of vector addition, the Mapping 2 has simple form, we shall throughout the thesis use Mapping 2. Another reason is that with Mapping 2, it is easy to identify the histogram patterns in the code spectrum. The transformation formula for the conversion of one type of mapping to another is given in [11].

2.3.1 Review of decoding algorithms for RM codes

In view of the fact that WHT based decoding algorithms are applicable to the class of RM codes, we undertake a brief review of various available decoding algorithm for RM codes, emphasizing their merits and demerits. This helps not only to motivate WHT based decoding algorithm but also to understand evolution of various decoding algorithms in the proper perspective.

A decoding algorithm for decoding $R(r,2^m)$, for any general r and m, was first proposed by Reed [19]. This decoding algorithm is the fore–runner of decoding algorithms that came to be known as "majority logic decoding algorithms". These decoding algorithms usually require several levels of majority logic decisions in order to completely decode to the minimum distance. However, an inherent disadvantage is that the complexity of the required logical circuitry grows rapidly both with the number of errors to be corrected and the number of majority logic levels employed. Subsequently, these decoding algorithms were improvised by the works of Weldon, Chen and Hartmann [61]–[63]. Rudolph showed that it is possible to simplify the decoding hardware at the expense of decoding delay [63].

In 1966 the work of Green in connection with Mariner space mission led to a simple correlating decoding algorithm for $R(1,2^m)$ based on WHT [5]–[6] – the first decoding algorithm to make use of WHT techniques. This algorithm was presented in an elegant form in 1980's by MacWilliams and Sloane [7]. Recently, Simon N. Litsyn in [64] discussed computationally efficient ways of implementing this algorithm. Another important contribution is that of Tokiwa et.al who gave a new decoding algorithm (abbreviated TSKN algorithm) based on the "superimposition property" of RM codes to decode upto minimum distance for $R(r,2^m)$ [46].

There are other contributions not of immediate concern to us. Seroussi and Lempel discussed maximum likelihood decoding algorithms of certain Reed–Muller codes based on properties of binary symplectic matrix, [65]. Another interesting contribution [66] employs WHT for the purpose of soft–decision decoding of RM codes. However, soft–decision decoding algorithm requires doing WHT of the order that is exponential in dimension 'k', which may not be practically attractive.

We present deconvolution approach to decoding of RM codes based on WHT domain techniques. The main motivation for WHT techniques is the simple nature of decoding operations such as comparison, substitution and integer additions. By judiciously using properties of RM codes with the WHT techniques, we may get decoding algorithms that are simple and efficient. Before presenting the decoding approach, we first consider WHT spectrum characterization of RM codes that plays a crucial role in deriving WHT domain decoding algorithms.

2.3.2 WHT characterization of RM codes

We start by considering the well known WHT spectral characterization of first order RM codes [7]. By definition of WHT, it is seen that the basis vectors of WHT matrix are code vectors of first order RM

code. It follows from orthogonality property of the basis vectors that the correlation of a codeword with any other codeword is zero, and therefore the code spectrum contains a single nonzero position of value $\pm 2^m$.

It is found that WHT spectrum of RM codes can be grouped into several histogram classes, where a histogram class consists of all transformed RM code vectors having the same distribution of spectral values. For computation of histogram structure only absolute values of the spectrum are considered. For example, the histogram structure of WHT spectrum of $R(1,2^m)$ consists of a single class, consisting of transformed code vectors having only one nonzero spectrum with absolute value 2^m; the rest of the (2^m-1) spectrum values being zero. To the best of our knowledge, there are no results on the WHT characterization of higher order RM codes. In Chapter 3, we show that WHT spectrum of $R(2,2^m)$ has $\lfloor m/2+1 \rfloor$ different classes, j^{th} class containing 2^{2j} nonzero spectral locations having value 2^{m-j} for $j = 0$ to $\lfloor m/2 \rfloor$.

In view of properties of WHT, it is seen that the histogram of the spectrum of a second order RM code vector can be interpreted as storing correlation values of a given code vector with respect to vectors of a distinguished subset of the code. This follows because first order RM codes are sub–codes of higher order RM codes. WHT characterization of RM codes then means the histogram structure observed in these correlation values of the codewords of higher order RM codes w.r.t code vectors of first order subcode.

For deriving the histogram characterization of second order RM codes it is necessary to invoke results concerning canonical representation of multivariable polynomials over binary fields. Unfortunately, it is not easy to determine histogram classes in the WHT spectrum of third and higher order codes primarily because of non–availability of appropriate canonical forms.

2.3.3 Deconvolution procedures for WHT domain decoding of RM codes

In time domain a code vector is transmitted as a sequences of ± 1, and the effect of error at any position is to change $+1$ to -1 or vice versa. For purposes of analysis, error can be modeled as an auxiliary vector containing ± 2 at error locations added to the code vector. In the transform domain, it is necessary to invoke addition property of WHT to analyze how the received word spectrum is modified from the transmitted codeword spectrum. This WHT spectrum analysis due to error modification is in general quite tedious to carry out completely except in case of small errors. Fortunately, it is found that the knowledge of range of variation of spectral values is useful for decoding purpose. In particular the maximum possible deviation of a spectral value is linearly related to number of errors occurred. This is a vital clue for recovering the transmitted codeword spectrum from the knowledge of histogram of the code spectrum.

Let us illustrate the method concretely for the case of $R(1,2^m)$. The d_{min} of $R(1,2^m)$ code is 2^{m-1} and the code can correct upto $2^{m-2}-1$ errors. It is known that the WHT spectrum of $R(1,2^m)$ consists of a

single nonzero location with the value $\pm 2^m$. Now let us consider the maximum possible change in spectral values. If a spectral location is "0" valued, then it can change from $-(2^{m-1}-2)$ to $(2^{m-1}-2)$. The corresponding deviation for maximal valued spectral location is from $2^{m-1}+2$ to 2^m. This means that if error multiplicity is within the error–correcting capability of the code, the position of maximal value spectral location does not change. The decoding operation then involves simple pattern recognition which, in the present case, is the operation of finding the location of maximal absolute valued spectrum. Therefore, the transmitted codeword is obtained by substituting maximal location with 2^m, keeping the sign intact, and replacing other locations with zeros and then performing inverse WHT. Note that the decoding process involves only simple operations of finding maximum location and of substitution in the WHT spectrum of the received vector.

We now proceed to outline the WHT based methodology for second and higher order RM codes. We have mentioned in Section 2.3.2 that second order RM codes have several histogram classes in the WHT spectrum. The analysis of modifications in the WHT spectrum, due to additive errors, is done similar to the case of first order code. The possibility of recovering the histogram of the transmitted word is based on the following two conditions: that a nonzero valued spectral location does not change sign even when it is modified by maximum possible deviation and that the range of variation of spectral values within a histogram class are distinct.

If the conditions for recovering histogram are satisfied the decoding proceeds as follows: compare each of the received spectrum with suitably chosen thresholds to determine the possible choices of transmitted spectrum. The number of received spectral components whose values lie in the range of variation corresponding to zero spectral value is determined. In most of the cases this count provides strong clues to identify the histogram class of the transmitted codeword. Once the histogram class is determined appropriate substitutions are made from the possible choices for a received spectrum to determine the transmitted codeword spectrum.

Analysis of histogram recoverability conditions in Chapter 3 shows that it is still not possible to decode $R(2,2^m)$ in a straightforward manner, the exception being $R(2,2^4)$, which is the smallest non–trivial single–error–correcting code of length 16. Therefore, it becomes necessary to use "superimposition property" of RM codes to decompose a given second order RM decoding into decoding of several subcodes which can be decoded by WHT methods. This approach is used in Chapter 3 for obtaining decoding algorithms for general second and third order RM codes.

2.4 DFT domain decoding approaches

Blahut, by bringing together various approaches in the earlier literature, clearly formulated a spectral methodology for understanding the theory of cyclic codes in [2]–[4]. The attractiveness of his viewpoint lies in bringing the coding theory subject in line with digital

signal processing. Finite field discrete Fourier transforms are made to play a vital role in this framework. Codewords can be considered as discrete time signals defined over a finite index set and taking values from a finite field. This interpretation implies that algebraic properties of the code are inherited by the signal space formed by the codewords of the code.

Both 1–D and 2–D DFT characterize cyclic codes as having specified set of locations identically equal to zero. In [59] mixed–radix representation of an abelian group is employed for characterization of 1–D and 2–D DFT which is elegant for computer implementation. Our main purpose in regard to DFT based spectral decoding algorithms is to present a unified framework to understand and design 1–D and 2–D decoding algorithms. We shall not only give an alternative framework but also apply the methodology to derive new decoding algorithms.

2.4.1 Review of DFT domain encoding and decoding algorithms for cyclic codes

DFT methods provide for alternative encoding and decoding procedures for the class of cyclic codes. The DFT based encoding methods for different codes are similar except for appropriate modifications depending on the defining zero locations of the code. The DFT code spectrum is such that only some specified components, namely conjugacy leaders, need to be independently chosen to specify a codeword. Once these are chosen, the remaining components are determined from the conjugacy relations that hold among the members of the conjugacy class. Each conjugacy class has specified "bit–content" associated with it. It is also referred to as order of the conjugacy class. Codes, by definition, have some conjugacy classes always zero. The total bit–content of remaining non–zero conjugacy classes is equal to the number of information bits. Hence the information bits can be made to specify the spectral value of nonzero conjugacy class leaders. Then the remaining obligatory frequency components are filled with appropriate powers according to conjugacy constraints and inverse transform is taken to obtain the codeword. At the time of decoding, after received vector is decoded to the correct codeword, information bits are directly read out from the nonzero spectral components. This type of encoding is non–systematic in nature. For the case of composite code lengths such a spectral encoder may be simpler than the corresponding time domain convolution encoder [2].

DFT based decoding techniques, besides adding to repertory of decoding techniques already available, can easily be modified for decoding of erasures and for decoding beyond the designed BCH bound. The important developments are improvisation of BCH decoding algorithm to correct beyond BCH bound and application of BCH decoding idea for decoding of generalized BCH codes (Alternant codes and Goppa codes) and multi–dimensional BCH codes. Another direction of research is applying DFT domain techniques to decoding of some non–BCH cyclic codes. We also review developments in other related areas such as generalizations and improvements of BCH lower bounds on minimum distance of cyclic codes and

advances in linear feedback shift register (LFSR) synthesis algorithms which are important subalgorithms of BCH decoding algorithm.

The concept and derivation of BCH bound on minimum distance has played a vital role in the development of DFT decoding algorithms. The BCH lower bound on minimum distance is given by the number of consecutive zero locations plus one. The method of derivation was instrumental to the development of Peterson's BCH decoding algorithm [12]. The BCH bound was generalized to Hartmann–Tzeng (HT) bound [67] and subsequently to Roos bound [68], both based on multiple set of consecutive roots. Recently, Van Lint and Wilson devised new techniques of bounding which includes earlier BCH bound, HT bound and Roos bounds on the minimum distance and applied it successfully to yield the true minimum distance for code lengths < 63 with minor exceptions [55]. Two dimensional extensions of BCH bound have been studied in [3]–[4].

Blahut has given spectral domain implementation of 1–D BCH decoding in [2]–[3] and discussed modification of the algorithm for errors–and–erasures decoding and for decoding beyond BCH bound. For decoding algorithms trying to correct beyond the BCH bound, the main disadvantage is that decoding computations increase rapidly with the number of extra errors to be corrected. Application of spectral domain BCH decoding idea to Goppa codes is treated in [9] and to alternant codes in [8]. The motivation for considering alternant and Goppa codes is that the minimum distance is much larger than the designed bound. Even though BCH decoding idea can decode upto the designed distance for both these class of codes, it has limited potential for exploiting full error–correcting capability for these codes.

Another application area is the decoding of multi–dimensional cyclic codes which are characterizable by n–D DFT techniques. Blahut has introduced the concept of a two–dimensional BCH code having a contiguous array of $2t \times 2t$ zeros and shown that the minimum distance of the code is at least $2t+1$ [3]–[4]. He has also given two spectral decoding algorithms for error–correction in 2–D BCH codes: one for correcting upto 't' random errors and the other for correction of certain types of burst errors. The decoding algorithm proposed requires as an important subalgorithm the Berlekamp–Massey procedure for synthesizing a LFSR for a given syndrome sequence along a row (column). This is followed by the recursive extension of syndromes along that row (column). Another feature of the decoding algorithm is that the syndrome computations for either all the rows or all the columns can be done in parallel.

LFSR synthesis algorithm is important in its own right and also as an important subalgorithm for BCH decoding algorithms. Berlekamp proposed an iterative algorithm for the 1–D LFSR synthesis [32]. Massey presented an alternate simplified version of the iterative algorithm [33]. Alternatively, LFSR synthesis problem can be formulated as finding the greatest common divisor (GCD) of two polynomials using Euclidean algorithm in a recursive procedure [69], [70]. The method of using Euclidean algorithm for decoding in [69] is different from the method discussed in [70]. A good summary of different approaches to

1–D LFSR synthesis is discussed in [71]. The latter reference also gives another way of using Euclidean algorithm to decode BCH codes. The equivalence between B–M algorithm and Euclidean algorithm is considered in [72]. This fact simplifies the choice of LFSR algorithm for implementation depending upon application requirements while developing decoding architectures [44]–[45].

An associated problem to a single sequence LFSR synthesis problem is that of synthesizing minimum length LFSR for generating prescribed multiple sequences (MLFSR). Feng and Tzeng have presented a generalized Euclidean algorithm (we refer to it as FT–1) for completely solving the multiple LFSR synthesis problem [52]. Subsequently they have come up with a more general algorithm for finding the smallest initial set of linearly independent columns in a matrix over a finite field, called fundamental iterative algorithm (FIA). This more general problem formulation includes multisequence problem as a special case. They also derived, through a refinement of FIA, a generalization of B–M algorithm (we refer to it as FT–2) for solving the multisequence LFSR synthesis problem [53]. They also applied the algorithm for decoding of cyclic codes correcting up to the HT bound and in some instances upto the Roos bound. Recently in [54] procedures for errors–and–erasures decoding of cyclic codes up to the HT bound using MLFSR synthesis algorithms are given.

Another direction of research concerns generalizing the concept of LFSR to two or more dimensions. Sakata developed necessary theoretical details and proposed an efficient algorithm for solving multi–dimensional LFSR synthesis [47]–[49]. Fitzpatrick and Norton have proposed an alternate method of generating n–dimensional LFSR's [51].

There are some important non–BCH cyclic codes having performance comparable to BCH codes and for which spectral decoding is possible. Unfortunately, none of the decoding algorithms developed are applicable to decoding any general class of cyclic codes. The best one can hope is to have a framework of rules and tailor them to the requirements of a particular code. Bours et.al, by employing Newton's identity in an interesting way, have come up with algebraic decoding algorithms for several codes [10]. Recently, Tzeng and Feng have employed nonrecurrent syndrome relations to give decoding algorithms for various cyclic codes [73].

In the next two sections we show that interpretation of a DFT based decoding algorithm as a deconvolution in DFT domain leads to identifying important decoding approaches from which one can understand all the above mentioned 1–D and 2–D DFT based decoding algorithms. Some of the existing decoding algorithms for 2–D BCH codes are reviewed in this light and modifications for efficient implementation are suggested. Several new decoding procedures for some non–BCH cyclic codes are developed as another application of these ideas.

2.4.2 Stages in DFT domain decoding

We treat both 1–D and 2–D decoding algorithms in a single framework. An analysis of a typical spectral decoding algorithm reveals that there are three stages in decoding computation: the error spectrum computation stage, the error location stage and the error evaluation stage. Any DFT based approach centers around the error location stage which involves the formulation and computation of error locator polynomials. For efficient computation in the error location stage and in the error evaluation stage the structure of zero locations of the code has to be utilized to good advantage. In what follows, we discuss some general aspects of these three stages of decoding computation common to any DFT domain decoding approach.

1)   Error spectrum computation: The error spectrum is alternatively referred to as syndrome in the coding literature. It is known that zero locations are the positions in the codeword spectrum which take values identically equal to zero and therefore error spectrum or syndrome consists of contributions solely due to the error. These positions provide a sort of "window" through which one looks into the error domain and tries to reconstruct error from whatever information is available. The objective of any decoding algorithm is to set up a system of convolutional equations involving syndromes in these positions to find appropriate error location polynomials. It is desirable that these positions have some structure that may facilitate in efficiently solving the system of equations. Then, for the class of codes having similar structure of zero locations the decoding procedure remains essentially the same.

2)   Error location: The purpose of this stage is to find the error locator polynomial which is formed by error position locators. This is the polynomial corresponding to Λ in (2.2.2), where Λ is DFT of the auxiliary vector λ. The importance of error locator polynomial is that it is the minimal order connection polynomial of the sequence formed by the syndromes. Therefore this polynomial is also referred to as connection polynomial or recurrence polynomial or row (column) connection polynomial (in 2–D case). The concept of error locator for 1–D case is just the locator corresponding to error positions. For the 2–D case two possibilities are possible: error locator corresponding to 2–D locations of errors (straightforward extension of 1–D concept) and error locator corresponding to row (column) positions of the error. These different concepts of error locator lead to different decoding approaches. The main computation problem in this stage of decoding is the inverse problem of finding the error locator polynomial given part of the syndrome array. This inverse problem is equivalent to synthesizing a LFSR problem. The Berlekamp–Massey algorithm requires a sequence of consecutive syndromes for efficient computation which translates to requiring continuous zero locations in the code definition.

3)   Error evaluation: This stage is the most computationally expensive step of the decoding. However, once the error locator polynomial is obtained there are several ways to complete the decoding in either

time domain or spectral domain. The selection of any one of these methods can be made so as to reduce the overall computational complexity.

Following spectral domain method, first recursive syndrome extension has to be performed, to determine the complete error spectrum vector and then perform inverse DFT to obtain the error [2]–[4]. Since the concept of syndrome extension is closely related to the formulation of error locator polynomial, the specific details regarding syndrome extension will be discussed along with respective decoding approaches.

For decoding binary codes in time domain the error locator polynomials are sufficient. The zeros of error locator polynomials are found by Chien search, as once an error position is identified, the remedy is to simply invert the bit. For the non–binary case the concept of error evaluator, introduced by Forney, is needed [31]. The B–M algorithm can be modified to compute error evaluator polynomial. However, finding the error values requires division unlike spectral method of recursive extension.

We have seen earlier that any DFT domain decoding algorithm is equivalent to zero output deconvolution involving Λ and E, assuming available partial knowledge of E. The nature of convolution of Λ and E is determined by the definition of error locator adopted in the error location stage. Therefore a particular DFT decoding approach is typified by the nature of deconvolution corresponding to the nature of convolution of Λ and E. We study two distinct DFT decoding approaches: 1–D deconvolution approach based on a 2–D linear complexity theorem and 2–D deconvolution approach based on Sakata's algorithm for 2–D LFSR synthesis. The former approach is our contribution based on a 2–D generalization of Blahut's 1–D linear complexity theorem [50].

2.5 1-D deconvolution approach based on complexity theorem

2.5.1 A 2-D generalization of linear complexity theorem

The notion of linear complexity is very important in coding theory. It is a versatile concept which is employed to derive lower bounds on minimum distance and to find weight distribution of codes. We want to highlight the role of the complexity theorem to understand decoding algorithms. Towards this end, in this subsection we obtain a useful 2–D generalization of Blahut's 1–D linear complexity theorem.

The following notations and definitions are needed

Definition 2.5.1 : Let p^N denote a sequence $(p_0, p_1, ..., p_{N-1})$ whose components belong to some specified field F. The linear complexity of p^N (denoted by LC) is defined as the smallest non–negative integer L such that there exist $a_1, a_2, ..., a_L$ in F for which

$$p_j + a_1 p_{j-1} + ... + a_L p_{j-L} = 0, \quad \text{for } L \leq j < N; \tag{2.5.1}$$

and as ∞ in case no such integer exists.

Definition 2.5.2 : The semi–infinite sequence $p^\infty = (p_0, p_1, p_2 \ldots)$ is said to be periodic with period N, if $p_j = p_{j+N}$. The smallest N for which the sequence is periodic is called the fundamental period.

A periodic sequence p^∞ is completely characterized by its first period $p^N = (p_0, p_1, \ldots, p_{N-1})$. The cyclic left–shift operator S on N–tuples is defined as $S(p^N) = (s_1, \ldots, s_{N-1}, s_0)$ and $S^2(p^N) = S(S(p^N))$ and so on. If p^∞ is a periodic sequence having period N, then (2.5.1) becomes

$$p_j + a_1 p_{j-1} + \ldots + a_L p_{j-L} = 0, \quad \text{for } L \leq j < N+L \tag{2.5.2}$$

This in turn is equivalent to

$$S^L(p^N) + a_1 S^{L-1}(p^N) + \ldots + a_L(p^N) = 0^N \tag{2.5.3}$$

The idea of complexity theorem for the 1–D case though implicit in Blahut's work [4], however, was first stated clearly by Massey and Schaub [50].

Theorem 2.5.1 : If B^N is the 1–D DFT of b^N and $W(b^N)$ denotes Hamming weight of b^N, then the linear complexity (LC) of sequences are given by

$$LC(b^\infty) = W(B^N) \text{ and } W(b^N) = LC(B^\infty) .$$

In order to render explicitly the role of connection polynomials we quote below the following theorem from [74] which can be considered as a generalization of Theorem 2.5.1. Let $p_0, p_1, \ldots$ be a k–th order homogeneous linear recurring sequence in a field F satisfying the linear recurrence relation

$$P_{n+k} = a_1 P_{n+k-1} + a_2 P_{n+k-2} + \ldots + a_k P_n \; ; n = 0, 1, 2 \ldots ,$$

where $a_j \in F; 0 \leq j \leq k-1$. The polynomial

$$f(x) = (x^k - a_1 x^{k-1} - a_2 x^{k-2} - \ldots - a_k) \in F[x]$$

is called the characteristic polynomial (alternate name for connection polynomial) of the given linear recurring sequence.

Theorem 2.5.2 : Let $p_0, p_1, \ldots$ be a kth order homogeneous linear recurring sequence in F with the characteristic polynomial f(x) as given above. If the roots $\beta_1, \ldots, \beta_v$ of f(x) are all distinct, then

$$P_n = \sum_{j=1}^{v} \alpha_j (\beta_j)^n \; ; \; n = 0, 1, \ldots$$

where $\alpha_1, \alpha_2, \ldots, \alpha_v$ are elements in F that are uniquely determined by the initial values of the sequence and belong to splitting field of f(x) over F[x].

Remark 2.5.1: Theorem 2.5.2 is more general than Theorem 2.5.1 as it is applicable to the class of closely related truncated DFT transforms. Such transforms are considered in the context of studying

characterization and decoding of Goppa codes [9] and alternant codes [8]. Theorem 2.5.2 shows that the roots of the connection polynomial give error locators.

Remark 2.5.2: It is to be noted that in the context of coding theory time–domain vector may be interpreted either as a code vector or as an error vector, as the case may be. For the case of transmission over noisy communication channels, it is usually the case that an error vector of smaller weight is more probable than that of larger weight. This fact translates, in the spectral domain setting, to the requirement that the linear complexity of the error transform sequence must be as small as possible.

Remark 2.5.3: 1–D complexity theorem is also applicable to the case of two–dimensional transform sequence in the following way. The two–dimensional transform sequence of a given 2–D array has several row (column) sub–sequences and each row (column) sub–sequence is periodic with period equal to the row (column) length. Suppose time domain pattern is

$$c(x,y) = \sum_{k=1}^{v} y^{j_k} c_k(x) \,,$$

where $c_k(x)$ are polynomials in x. Then the transform coefficients of i–th row are given by

$$C_{i,j} = \sum_{k=1}^{v'} \beta^{jj_k} c_k(\alpha^i) \;;, \quad j = 0, 1, ..., n-1$$

where $v' \leq v$ and $c_k(\alpha^i) \neq 0$ for k = 1 to v'. From Theorem 2.5.2 it follows that the degree of the connection polynomial is equal to the number of nonzero $c_k(\alpha^i)$ for k = 1 to v. Therefore the linear complexity of the row–sequence can be less than the number of nonzero columns.

Remark 2.5.4: It follows from Remark 2.5.1 that for the 1–D cyclic decoding case the roots of the connection polynomial of the error transform sequence give the locations of errors. Likewise, it is clear from the above discussion that in the 2–D cyclic decoding case, the row (column) recurrence polynomial has a similar interpretation. If β^i is a root of the row recurrence polynomial, then i–th column is corrupted by the error. Similarly the roots of the column recurrence polynomial give the rows corrupted by the error.

From Remarks 2.5.3 and 2.5.4 , it is evident that the generalization of complexity theorem to 2–D requires some modifications. Given a 2–D sequence there are several row or column sub–sequences and consequently the minimal degree connection polynomial generating all the row or column sub–sequences is of interest. The minimal degree of this connection polynomial is defined as the row–complexity of the 2–D sequence. Similarly the column–complexity of a given 2–D sequence is defined. The following theorem is a generalization of Blahut's complexity theorem to two dimensions.

Theorem 2.5.3 : Let c be a two dimensional array and C be its DFT. Then the row (column)–complexity of the LFSR generating all the row (column) sub–sequences of C is equal to the number of nonzero columns (rows) of c.

Proof. Let $c(x,y) = \sum_{k=1}^{n(c)} c_k(x) y^{j_k}$ be the time domain polynomial where n(c) denotes the number of

nonzero columns. Then i^{th} row of transform sequence is $C_i^N = \{C_{i,j} = \sum_{k=1}^{n(c)} \beta^{jj}_k c_k(\alpha^i); j = 0 \text{ to } n-1\}$. We have

$$p(S)\ (C_i^N) \triangleq \sum_{k=1}^{n(c)} c_k(\alpha^i)\ B_{j_k}^N\ p(\beta^{jk})\ ,$$

where $B_{j(k)}^N = (1,\ \beta^{jk},\ \beta^{2jk}, ...,\ \beta^{(N-1)jk})$ and $p(S)$ is the vector shift notation explained before. If $c_r(z)$ stands for $\prod_{k=1}^{n(c)} (z-\beta^{jk})$, then it is simple to check that $p(S) = c_r(S)$ satisfies all the row sub–sequences. Our claim is that $c_r(z)$ is the least degree polynomial having such a property. If possible, let $c_r(\beta^{jt}) \neq 0$ for some column index $t \in \{1, ..., n(c)\}$ and for some row, say i. We shall now show that

$$c_r(S)(C_i^N) = \sum_{k=1}^{n(c)} c_k(\alpha^i)\ B_{j_k}^N\ c_r(\beta^{jk}) \neq 0 \qquad (2.5.4)$$

Multiplying (2.5.4) by $q(S) = \prod_{k \neq t} (S-\beta^{jk})$ for $k = 1,..,n(c)$, it is seen that $q(S)c_r(S)(C_i^N)$ simplifies to a single term $c_t(\alpha^i)\ B_{j_t}^N q(\beta^{jt})\ c_r(\beta^{jt})$ which is not zero as $q(\beta^{jt}) \neq 0$ by construction and $c_r(\beta^{jt}) \neq 0$ by hypothesis. Therefore

$$q(S)c_r(S)(C_i^N) \neq 0 \qquad (2.5.5)$$

is proved. This implies (2.5.4), for otherwise, we obtain contradiction to (2.5.5). As (2.5.4) is obtained starting with the assumption $c_r(\beta^{jt}) \neq 0$, in order to make $c_r(S)(C_i^N)$ zero, β^{jt} has to be a root of $c_r(z)$. In other words it is proved that the the degree of $c_r(z)$ is minimal. □

Remark 2.5.5: When a 2–D array consists of only one row, i.e. corresponding to 1–D case, Theorem 2.5.3 reduces to Blahut's 1–D linear complexity theorem as the number of nonzero columns is equal to the Hamming weight. However, for 2–D case the product of the row complexity and the column complexity upper bounds the Hamming distance of the corresponding time domain vector. Similar to 1–D case the roots of the common row (column) polynomial considered in Theorem 2.5.3 are interpreted as column (row) locators of the nonzero column (row) positions.

Remark 2.5.6: An important feature of the generalization is that row–complexity is no longer directly related to the Hamming weight as in the 1–D case. Yet this is not disadvantageous as, on one hand, it allows for correction of some burst–errors, and, on other hand, it simplifies computations in some cases, as row–complexity may be less than Hamming weight of the error pattern due to grouping of errors.

2.5.2 1–D deconvolution approach for decoding of cyclic codes

The basic idea of this approach is to associate a nonzero error position (i,j) with a 1–D error–locator corresponding to either row index i (α^i) or column index j (β^j). The error locator polynomial,

in this formulation, is formed by the rows (columns) corrupted by error. Therefore this approach is a 2–D generalization of 1–D decoding formulation. From Theorem 2.5.3 it follows that 1–D convolution of the common row (column) error locator with each of the row (column) subsequences of the 2–D error spectrum is zero. This shows that a typical 2–D decoding problem can be formulated as equivalent to several deconvolution problems which for the DFT case are equivalent to synthesizing several 1–D LFSRs corresponding to the row or column subsequences. An advantage of this approach to 2–D decoding is that it is possible to use architectures developed for B–M algorithm or its equivalent Euclidean version to develop decoding architectures with little modifications.

There are various choices for LFSR synthesis algorithms: Berlekamp–Massey algorithm [32]–[33], Euclidean algorithm [69]–[71], and recently proposed Feng–Tzeng algorithms for multiple LFSR synthesis. [52]–[54]. For 1–D cyclic codes B–M algorithm is intended for correcting upto BCH bound whereas FT algorithm is intended for correction upto Hartmann–Tzeng bound for cyclic codes. For 2–D codes we shall see that decoding can be formulated by either of these LFSR algorithms with different error correcting capabilities. Then the choice of the LFSR algorithm may be made from the noise characteristics of the channel.

We now describe error evaluation based on recursive extension method for 2–D codes. It is well–known that for 1–D case recursive extension of syndromes using the connection polynomial proceeds in a straightforward way either in the forward direction or in the backward direction. For a 2–D code having area $n_1 \times n_2$, there are several possibilities to carry out recursive extension. The usual way is to do recursive extension along the rows and columns. Other possibilities include recursive extension along the diagonal row (column) or skewed row (column) or skewed diagonal row (column) and so on. The period of a row (column) syndrome sequence thus generated is a divisor of $LCM(n_1, n_2)$. If $LCM(n_1, n_2) = n_1 n_2$, then all the syndromes are computed in one recursion step. Therefore such recursive procedures combined with computationally efficient 2–D transforms are employed for fast decoding of some class of BCH codes where n_1 and n_2 are relatively prime [3]–[4].

In what follows, decoding formulations of BCH codes having simple zero location structure are interpreted in the framework of 1–D deconvolution decoding approach. First example considers the formulation of spectral decoding for 1–D BCH codes. The next two examples discuss two different spectral decoding formulations for the class of 2–D BCH codes.

Example 2.5.1 : Consider the formulation of decoding problem of 1–D BCH code of length n having a continuous array of 2t zeros. The designed distance of the code is 2t+1 and therefore correction upto 't' random errors is possible. Suppose $r = c \oplus e$, is the received vector and e is the error vector and R, C and E, denote the corresponding spectral domain vectors. If $v \leq t$ error happen the minimal complexity of the

error sequence is upper bounded by t. Since 2t consecutive syndromes are available the desired connection polynomial is found from B–M algorithm. The syndrome extension of these 2t consecutive syndromes determines the complete error syndrome array. An inverse transformation of (R–E) completes the decoding.

Example 2.5.2: Consider a 2–D BCH code of dimensions n × n. The defining zeros of the code, in the form of a 2t × 2t contiguous array, are given by $C_{j,j'} = 0$, for j = 1 to 2t and j' = 1 to 2t. Consider the case of correction of v ≤ t random errors. The number of rows or columns corrupted do not exceed 't' and therefore the minimal complexity of any row (column) sub–sequence remains upper bounded by t. Hence 2t consecutive syndromes in a row (column) produce the required connection polynomial using B–M algorithm. After 2t rows have been processed, all the syndromes in the first 2t rows (columns) are known. Then the B–M algorithm followed by recursive extension is repeated in the column (row) direction to determine the complete error syndrome. Then inverse transform of R–E completes the decoding.

Example 2.5.3: For the code considered in the Example 2.5.2, an alternate decoding procedure is derived as follows. The procedure is to find directly the common row (column) connection polynomials whose complexities for the given error pattern of t random errors remain upper bounded by t. The problem of finding the common row connection polynomial can be viewed as a multisequence shift–register (abbreviated MLFSR) synthesis [52]–[54]. It will be shown later that MLFSR based decoding algorithm can correct some two–dimensional burst–errors which are not correctable by the Blahut's decoding algorithms.

2–D BCH codes are a 2–D generalization of 1–D BCH codes having a simple structure of zero locations in the form of a contiguous array of area 2t × 2t. From the definition of error locators it is clear that the 2–D span of locations corresponding to product of zeros of row connection polynomial and column connection polynomial contains all the error positions. An important observation is that 1–D deconvolution approach to decoding seems elegant to handle 2–D burst–errors. This interpretation is obtained as direct application of 2–D complexity theorem. In Chapter 4, we shall use complexity theorem to analyze the error–correcting capabilities of these 2–D BCH codes as afforded by the defining zero locations. This gives a performance index measuring error–correcting capability of the code against decoding capability of the decoding algorithm.

It is in the application of the methodology to random–error correction of 1–D and 2–D cyclic codes (of course it is assumed that 1–D codes considered admit of 2–D spectral characterization) that the role of complexity becomes more transparent. The decoding problem may again be viewed as synthesizing several one dimensional LFSRs corresponding to row or column syndromes. Because the error pattern is

interpreted as a 2–D error pattern, certain groupings of error take place, and as observed earlier in Remark 2.5.6, row complexity may be less than the Hamming weight of the error pattern. For any two row syndrome sub–sequences, it is possible that roots of the connection polynomial of one row may give partial information about errors in another row syndrome sequence. This partial information on error helps reduce the requirement on the number of zero locations to find the net row complexity of the second row sequence. This is a general explanation of how it is possible to decode some non–BCH cyclic codes using 2–D spectral techniques. In order to develop the decoding algorithm it is necessary to analyze different configurations of errors upto $[d_{min}-1/2]$. Usually the complexity of a distinguished row syndrome is found and used as guide to anticipate different possible configurations and proceed accordingly.

2.6 2–D deconvolution approach based on Sakata's algorithm

In this section we discuss an alternative approach to DFT decoding problem centered around Sakata's algorithm for 2–D LFSR synthesis. The basic idea of this approach is to associate an error position (i,j) with a 2–D error locator (α^i, β^j), corresponding to row and column indices of the error position. This is a straightforward extension of the 1–D concept of the error locator. As a first step towards 2–D decoding formulation, we now show that 2–D convolution of the transform vectors corresponding to the error locator and error syndrome polynomial is zero.

Let Σ_0 be the set of 2–tuples $m = (m_1, m_2)$, where m_1 and m_2 are non–negative. Consider a 2–variable codeword polynomial c(x,y) which is characterized by zeros $C_{i,j} = c(\alpha^i, \beta^j) = 0$, $(i,j) \in U$, a subset of Σ_0. If $r(x,y) = c(x,y) + e(x,y)$ is the received word, then we have

$$R_{i,j} = C_{i,j} + E_{i,j}$$

$$E_{i,j} = \sum_{k=1}^{t} e_{i_k, j_k} \alpha^{i i_k} \beta^{j j_k} \, ; \, (i,j) \in U$$

Consider a t–random error pattern with error locations at (i_k, j_k) for k = 1 to t. Define a 2–D auxiliary polynomial $\lambda_{i_k, j_k} = 0$, for k = 1 to t. Forming direct product with error vector e, we have

$$\lambda . e = 0 \tag{2.6.1}$$

Taking the 2–D DFT of (2.6.1) we obtain

$$\Lambda * E = 0$$

or

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \Lambda_{i,j} E_{v-i, w-j} = 0, \text{ for } (v,w) \in \Sigma_0^q \tag{2.6.2}$$

where n x n is the area of the code. In (2.6.2) the 2–D vector Λ corresponds to 2–D error locator polynomial $\Lambda(x,y)$ which has zeros at $(\alpha^{-i_k}, \beta^{-j_k})$, for k = 1 to t, analogous to 1–D case. The degree of

$\Lambda(x,y)$ is (p,q) where $p,q \leq t$. In order to be consistent with the convention for error locator polynomial followed by Sakata [47]–[49], some restructuring of Λ in (2.6.2) is necessary. Defining $\Gamma_{i,j} = \Lambda_{p-i,q-j}$, we have

$$\sum_{i=0}^{p-1} \sum_{j=0}^{q-1} \Gamma_{i,j} E_{i+v-p,j+w-q} = 0, \text{ for } (p,q) \leq (v,w) \in \Sigma_0^q \tag{2.6.3}$$

The next step in the 2–D decoding formulation is how to obtain a simple and efficient procedure for 2–D deconvolution. Towards this end it becomes advantageous to look at the algebraic structure of all possible 2–D error locator polynomials $\Gamma(x.y)$ satisfying (2.6.3). It is simple to check that the set of $\Gamma(x.y)$ is an ideal in $F_q[x,y]$, where F_q is the field in which syndromes are defined. This ideal I_e is generated by

$$I_e = \prod_{k=1}^{t} (x-\alpha^{i_k}, y-\beta^{j_k})$$

The generating set of ideal I_e satisfying (2.6.3), with the additional property that the polynomials in the generating set be of minimal degree w.r.t. a chosen 2–D total order, to be defined below, is of interest. This basis goes by the name of *Groebner basis* of the ideal and the basis elements are desired connection polynomials. Sakata has developed algorithms extending concepts of Berlekamp–Massey algorithm for LFSR synthesis to two dimensions to obtain 2–D LFSR synthesis algorithm in [47]–[48] and subsequently algorithms for n–D LFSR synthesis problem in [49]. It is to be noted here that the generalization of B–M algorithm by Sakata is of a different algebraic nature compared to generalization of B–M algorithm considered by Feng and Tzeng [52]–[54]. When reduced to one dimension, the decoding formulation reduces to well–known BCH formulation.

In view of this, 2–D error locator polynomials are also called 2–D connection polynomials. The 2–D deconvolution problem is therefore equivalent to 2–D LFSR synthesis problem. This problem is solved based on Sakata's algorithm for obtaining 2–D LFSR with minimum storing elements. The structure of zero locations is so chosen that the uniqueness of the set F of connection polynomials is assured [75]. Once set F is obtained, the error locators are found solving the system of equations represented by F. Error evaluation stage is completed by 2–D recursive extension. For the binary case, an alternate time domain correcting procedure is possible by computing the zeros of error locator polynomials and inverting the received bits at corresponding locator positions.

We have seen that in 1–D convolution approach the nature of syndrome definition is multi–dimensional whereas LFSR synthesis is essentially treated as one dimensional whether generating complete syndrome sequence or a periodic column or row sub–sequence. The 2–D deconvolution approach differs from 1–D deconvolution approach in viewing both the syndrome definition and the LFSR synthesis as 2–D.

2.6.1 A heuristic explanation of Sakata's algorithm

We now heuristically describe the inner working of the Sakata's 2–D LFSR synthesis algorithm [47]–[48]. We require following notations. Over Σ_0 define partial order "$\preceq$" defined as

$$(m_1, m_2) \preceq (n_1, n_2) \text{ iff } (m_1 \leq n_1) \wedge (m_2 \leq n_2).$$

where $\wedge$ is logical "and" operation. We introduce another total order "$\preceq_T$" as

$$(m_1 < m_2) \preceq_T (n_1, n_2) \text{ iff } (m_2 < n_2) \wedge ((m_2 = n_2) \vee (m_1 \leq n_1)) ,$$

where $\vee$ is logical "or" operation.

Sakata's algorithm needs a sort of one–dimensional indexing scheme for a 2–D syndrome array. The algorithm proceeds by accessing the next element of the array and processing it to compute the desired set of recurrence polynomials. The indexing scheme is given below.

If $m = (m_1, m_2) \in \Sigma_0$, then the next point $m+1$ w.r.t. $\preceq_T$ is defined as

$$m+1 = (m_1-1, m_2+1) \text{ if } m_1 \geq 1$$

$$m+1 = (m_1+1, 0) \text{ if } m_1 = 0.$$

The numbering scheme then is as shown in the Figure 2.6.1.

We also need definition of the following subset Σ_s^q.

$$\Sigma_s^q = \{q \in \Sigma_0 \mid s \leq q <_T p\}.$$

$$
\begin{array}{llll}
0 & 2 & 5 & 9 \\
1 & 4 & 8 & \\
3 & 7 & & \\
6 & & &
\end{array}
$$

Figure 2.6.1   Numbering scheme corresponding to points of Σ_0.

A 2–D array $S=(s_p)$ is defined as a mapping from Γ_S, a subset of Σ_0, into a field K. We associate with a 2–D LFSR a bivariate polynomial $f(x,y) = \sum_{r \in \Gamma_f} f_r x^{r_1} y^{r_2} \in F[x,y]$, where, $\Gamma_f = \{i \in \Sigma_0 \mid f_i \neq 0\}$ and $r = (r_1, r_2)$. The degree of $f(z)$ is said to be maximum element of Γ_f w.r.t total order Σ_0 defined above.

For a polynomial of deg $(f) = t$, f is said to be valid at point q w.r.t array S iff

$$f[S]_q = \sum_{r \in \Gamma_f} f_r s_{r+q-t} = 0 \; ; q \in \Sigma_s^p \text{ or } q \preceq_T s.$$

This is then conveniently written as f[S]=0;

The set of valid polynomials form an ideal of the polynomial ring K[z]. This ideal, called VALPOL(S), is defined as

$$VALPOL(S) = \{f \in K[z] \,|\, f[S] = 0\}$$

For the cases of 2–D decoding problem, S will be an periodic array and then VALPOL(S) is an ideal. A finite subset F which generates VALPOL(S) is of interest. These set of polynomials represent 2–D counterpart of the 1–D recurrence polynomial. We also require that the polynomials of F to be of minimal degree w.r.t ordering $\preceq_T$. Sakata has shown that such a requirement is equivalent to saying that F is a Groebner basis and vice versa [47].

Sakata's main idea was to compute iteratively the recurrence relations based on a 2–D extension of B–M algorithm to obtain the required set of connection polynomials. Let F represents the connection polynomial set which satisfies recursion until most recent point. Then the next point of the array according to the indexing scheme of Fig. 2.6.1 is taken and the discrepancy is computed. Then we have 2–D counterpart of Berlekamp procedure where discrepancy term is utilized to compute new connection polynomial set F using previously stored auxiliary polynomial set. The auxiliary polynomial set were in turn obtained from connection polynomials of previous iterations. At each iteration, in addition to updating the connection polynomial set, it is also necessary do update auxiliary polynomial set.

2.6.2 Requirements on the structure of zero locations

We now consider the requirements this algorithm makes on the structure of the zero locations. Sakata has given a procedure for obtaining the zero location configurations for any number of errors and the configurations for the case of two and three errors are shown in the Fig. 2.6.2 [75]. We like to note that this method of finding the configuration gets cumbersome as the number of errors increases. For the general case the structure of zero locations seems to take the form of a triangle. This conjecture is supported by the definition of a narrow–sense 2–D BCH codes, introduced by Sakata in [75], as having zero locations in the form of a continuous array, where the consecutiveness is understood in the sense 1–D indexing scheme described in Section 2.6.1. Sakata does not give the error–correction capabilities for the class of narrow sense 2–D BCH codes but notes that these codes could be decoded based on 2–D LFSR algorithm [75].

In chapter 4 we shall study requirements on the structure of zero locations, when decoded by 1–D deconvolution approach, for the class of 2–D BCH codes for both random and burst error correction. For the present, we consider the following decoding example and solve it by both the methods, and make some observations.

i/j	0	1	2	3
0	Z	Z	Z	Z
1	Z	Z		
2	Z			
3	Z			

Figure 2.6.2a Structure of zero locations for double error–correction

Z represents zero locations

i/j	0	1	2	3	4	5
0	Z	Z	Z	Z	Z	Z
1	Z	Z	Z			
2	Z	Z				
3	Z					
4	Z					
5	Z					

Figure 2.6.2b Structure of zero locations for triple error–correction

Example 2.6.1: Consider the following 2–D code of length 5 × 5 capable of correcting 2 errors. This example is taken from [75].

i/j	0	1	2	3	4
0		Z	Z	Z	Z
1	Z	Z			
2	Z		Z		
3	Z			Z	
4	Z				Z

Figure 2.6.3. Zero locations of (25, 13, 5) code

Suppose error locations are at $\{(0,2), (1,3)\}$. In order to correct 2 errors using Sakata algorithm we need to know the value of $S_{0,0}$. Because the code is binary $S_{0,0}$ can be 1 or 0. So performing Sakata's algorithm with the assumption that $S_{0,0}=0$ we end up with $F= \{x^2 + \theta^{14}x + \theta^3, y + \theta^6 x\}$. Solving for error locations we have $(0,2)$ and $(1,3)$ as answers. The other assumption $S_{0,0}= 1$ leads to non–minimal degree recurrence polynomials and therefore first choice of $S_{0,0}$ is preferable.

Decoding based 1–D deconvolution approach: Perform B–M algorithm on row syndromes $S_{0,1}$ to $S_{0,4}$ and obtain the row connection polynomial. If the degree of this connection polynomial is "0", obtain the single degree row connection polynomial from 2 consecutive syndromes in row 1. For the particular case, the row

connection polynomial is $(1 + \theta^{14}x + \theta^3 x^2)$ (Note B–M algorithm produces the connection polynomial in a different convention from that of Sakata's algorithm) and the column connection polynomial is $(1 + \theta^5 y + y^2)$. As for syndrome extension, the first two rows pose no problem and once the first row syndromes are known, other row syndromes are known by conjugacy property. ☐

Remark 2.6.1: First, we have Example 2.6.1 which shows that for the same given configuration of zero locations, 1–D deconvolution approach does not have to assume any extra information compared to 2–D deconvolution case. Secondly, the difference between these two approaches is more clearly illustrated for burst error correction. 2–D deconvolution approach formulates decoding problem on the basis of individual error locations whereas 1–D deconvolution approach formulates decoding on the basis of span of the burst error locations. As the span of a error pattern has less information content compared to entire error pattern, this may imply milder constraints on the structure of zero locations for decoding in 1–D deconvolution compared to 2–D deconvolution approach. Therefore it is not practical to use the 2–D deconvolution approach for the correction of burst errors. These two situations show the flexibility of 1–D deconvolution approach to handle different error configurations which comes in handy when decoding for cyclic codes is considered. For these reasons we prefer to use with 1–D deconvolution based approach for further investigations.

2.6.3 Interrelations between connection polynomials of 1-D and 2-D deconvolution approaches

It is of theoretical interest to know how the connection polynomials obtained in the 1–D deconvolution approach are related to connection polynomials obtained in 2–D deconvolution approach. To answer this question we invoke important results of Fitzpatrick and Norton in connection with alternative method for synthesizing n–D LFSR [51].

We consider the ideal generated by connection polynomials in these two approaches. In the 1–D deconvolution approach, it is clear from 2–D complexity theorem that the roots of the row (column) connection polynomial are locators of columns (rows) affected by error. Note that the product of zeros of the row and column connection polynomials correspond to 2–D rectangular span containing all error positions. There may be two different error patterns having the same span. The ideal of connection polynomials I_c in the 1–D deconvolution approach is generated by these minimal row and column connection polynomials. For a t–random error having zeros at (i_k, j_k), for k = 1 to t, the ideal I_c generated is given by

$$I_c = \left(\prod_{k=1}^{t}(x - \alpha^{i_k}), \prod_{k=1}^{t}(y - \beta^{j_k})\right)$$

2–D deconvolution approach starts from considering ideal generated by individual error locators and 2–D connection polynomials obtained by 2–D LFSR synthesis algorithm can be algebraically solved to get individual error locations. For the above example the ideal of 2–D connection polynomials I_e is given by

$$I_e = \prod_{k=1}^{t} (x - \alpha^{i_k}, y - \beta^{j_k})$$

It is clear that the that I_c is a sub ideal of I_e. This relation is expected from the definition of error locator used in these two cases. An algorithm called א–BASE is given in [51] to find generators of I_e from generators of I_c and some knowledge of error syndromes. א–BASE algorithm forms an important subalgorithm of an alternate 2–D LFSR synthesis algorithm proposed in [51]. Though א–BASE algorithm is presented for n–D case, for simplicity we consider only 2–D case.

The 2–D LFSR synthesis algorithm given in [51] assumes degree bounds on the row (column) connection polynomials and certain beginning terms of the 2–D sequence are known. In fact, the structure of locations of beginning terms of the 2–D sequence according to B–M hypothesis of [51] are similar to the structure of zero locations of 2–D BCH codes. The first step of 2–D LFSR computation is to find the row and column minimal polynomials using 1–D B–M algorithm or its equivalent. The remaining steps of the LFSR synthesis algorithm depends on the following characterization theorem concerning the connection polynomials generating the given 2–D array.

Theorem 2.6.1 : [Theorem 3.7, [51]] Let (σ) be a 2–D LFSR whose representation in terms of polynomial is given by p*G=q, where p is minimal product polynomials 2–D and G is the polynomial representation of (σ). Let $I(\sigma)$ be the ideal of connection polynomials. Then the following are equivalent

  a) $f \epsilon I(\sigma)$

  b)   there exists polynomials $u_k(X)$ such that $fq^* = \sum_{k=1}^{2} u_k p_k$

where, q^* denotes the reciprocal of a 2–variate polynomial q defined as $q^*(x_1, x_2) = x_1^{d_1} x_2^{d_2} q(1/x_1, 1/x_2)$, where d_i is the degree of q in x_i. The minimal product polynomial is defined as the product of row connection polynomials and the column connection polynomial.

The next in LFSR computation step is to complete the recursive extension of (σ) and compute the polynomial "q". It follows from the theorem that the desired set of recurrence polynomial is the Groebner basis of the set of polynomials satisfying condition (b). At this point א–BASE algorithm computes the Groebner basis of $I(\sigma)$ making use of Buchberger's algorithm [76].

Remark 2.6.2 : We note that as the knowledge of row (column) minimal connection polynomials is sufficient for the purpose of decoding of 1–D and 2–D codes, the present method of computing 2–D LFSR

is not of interest from the decoding viewpoint. However, this algorithm establishes the necessary theoretical link connecting these two approaches. We close this section by giving an example.

Example 2.6.2 : Consider the example of 2.6.1. The minimal polynomials in this case are given by $p_x = (x^2 + \theta^{14}x + \theta^3)$ and $p_y = (y^2 + \theta^5 y + 1)$. The polynomial q^* is found to be $(\theta^{11}y + \theta^2 x + \theta)$. The generators of the ideal are found to be $(x^2 + \theta^{14}x + \theta^3)$ and $(y + \theta^6 x)$ which are the same connection polynomials found by Sakata's method. $\qquad\square$

Chapter 3

SPECTRAL DOMAIN DECODING OF REED–MULLER CODES

In this chapter we consider a novel way of using spectral techniques based on Walsh–Hadamard Transforms (WHT) to develop decoding algorithms for the important class of Reed–Muller (RM) codes. The decoding algorithms can be interpreted as WHT domain deconvolution algorithms as explained in Chapter 2. Properties of WHT and RM codes are exploited to obtain simple decoding (deconvolution) procedure. The decoding operations include computation of WHT, comparison and substitution, which involve only integer arithmetic. These decoding operations lead to simpler implementation compared to DFT based decoding algorithms which require special purpose modules for doing Galois field arithmetic. Another attractive feature of WHT domain decoding approach is the possibility of faster computing of WHT using butterfly algorithm similar to the faster way of computing the DFT.

The class of RM codes are a sub–class of linear binary codes covering a wide range of rate and minimum distance [1], [7]. However, the error–correcting capabilities of the class of RM codes, except for first–order RM codes and RM codes of modest block length, are inferior to that of BCH codes. Their popularity rests on the fact that they can be decoded by simple majority logic. The decoding circuit complexity is moderate and fits into the scheme of Meggit's decoder architecture developed for cyclic codes. This has made the choice of choosing RM code competitive in some situations. Another interesting observation is that the class of first order RM codes are among those linear codes for which maximum likelihood decoding is practical and that the decoding algorithm is based on WHT [5]–[6]. First order RM codes have application in range–finding, synchronizing, modulation and scrambling. Historically, first order RM codes were first employed to transmit pictures from Mars. Another application of RM codes as inner codes in a concatenated coding systems is studied in [43].

Section 3.1 reviews the method of generation of RM codes and their properties. A general method of encoding r^{th} order RM code is given based on binary Reed–Muller transform. In Section 3.2, decoding algorithm for $R(r,2^m)$ due to Tokiwa et.al (which we call TSKN algorithm) is reviewed. The motivations for introducing the new WHT based decoding algorithm are explained. In Section 3.3, we digress to discuss some of the properties WHT spectrum of codewords of second order RM codes in detail. It is shown that $\lfloor m/2 \rfloor + 1$ different spectral classes can be identified in the WHT spectrum of $R(2,2^m)$. Some of the difficulties of extending these techniques to higher order RM codes are discussed.

Section 3.4 gives a new WHT domain decoding algorithm for $R(2,2^4)$ to correct single errors. This new decoding algorithm is used to obtain WHT domain decoding algorithm for $2^{m-4} * R(2,2^4)$. The latter decoding algorithm is utilized in decoding algorithm for $R(2,2^m)$ for any general m.

Section 3.5 starts with an alternative way of decomposing a general $R(2,2^m)$. Then we proceed to give a decoding algorithm for $R(2,2^m)$ to correct upto the error—correcting capability. The decoding algorithm uses two WHT subalgorithms: one for decoding of first order RM subcode and the other for decoding of $2^{m-4}*R(2,2^4)$ given in Section 3.4. Two variations of the WHT domain decoding algorithm for $R(2,2^m)$ are discussed.

In Section 3.6, we extend WHT domain techniques to give decoding algorithms for $R(3,2^m)$. First, a WHT decoding algorithm for $R(3,2^5)$ is obtained making use of computer results on WHT spectrum of $R(3,2^5)$ which is used to derive a WHT based algorithm for $2^{m-5}*R(3,2^5)$. The latter WHT decoding algorithm along with WHT decoding algorithm for $R(2,2^m)$ are employed for decoding of $R(3,2^m)$.

In Section 3.7, we give a schematic of decoding architectures for WHT domain decoding algorithm for second and third order RM codes. Then a review of delay for TSKN algorithm is given. An estimate of decoding delay for WHT decoding for second and third order codes is given. It is shown for $m \geq 8$, WHT decoding algorithm has less delay compared to TSKN algorithm.

3.1 Generation and properties of RM codes

Reed—Muller codes of length n ($n = 2^m$) are easily defined in terms of Boolean functions of $x_0 = 1$, x_1, x_2,..., x_m and their products. Truth—tables of these functions may be represented as binary vectors of length n ($n = 2^m$) whose components define function values at all possible input combinations. The order of tabulation corresponds to natural ascending order of minterms. Let v_i denote the vector corresponding to polynomial x_i. Then v_0 represents a binary function whose all 2^m components are 1's, and v_1, v_2, ... , v_m are such that when written as rows of a matrix have all possible m—tuples as columns. Define the vector product of two binary vectors as pointwise product of the respective components. Consider the collection of vectors formed by multiplying the vectors v_i ($i = 1$ to m) taken two at a time, three at a time and so on. The set of vectors so formed are linearly independent. Then r^{th} order RM code is the vector space spanned by the vectors v_0, v_1, v_2, ..., v_m and all vector products of these vectors taken r or fewer at a time. It follows therefore that there are $1 + {}^mC_1 + {}^mC_2 + ...+ {}^mC_r$ basis vectors in the rth order RM code. The minimum distance of the code is 2^{m-r}.

Example 3.1.1 : Table 3.1.1 shows all 16 basis vectors of RM code of length 16. The vector space spanned by first 5 vectors is the first—order (16, 5, 8) RM code. The vector space spanned by first 11 vectors is the second order (16, 11, 4) RM code; the vector space spanned by first 15 vectors is the third order (16, 15, 2)

Table 3.1.1 Generator matrix of Reed–Muller codes of length 16

v_0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
v_1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
v_2	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
v_3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
v_4	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
$v_1 v_2$	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
$v_1 v_3$	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$v_1 v_4$	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$v_2 v_3$	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$v_2 v_4$	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
$v_3 v_4$	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
$v_1 v_2 v_3$	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
$v_1 v_2 v_4$	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$v_1 v_3 v_4$	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$v_2 v_3 v_4$	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$v_1 v_2 v_3 v_4$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

RM code; and finally all 16 vectors span entire space of 16 tuples over GF(2). The basis vectors constitute generating codewords in the familiar generator matrix approach to encoding of RM codes.

A General Non–systematic Encoding of rth order RM Codes: It is well–known that any standard binary function expressed in terms of Reed–Muller basis functions 1, x_1, x_2, $x_1 x_2$,..., $x_1 x_2 ... x_m$ [77]. This suggests an elegant way to implement the encoding with the generator matrix discussed in Example 3.1.1. The idea is to extend the information vector to length n, where certain pre–assigned positions are set to zero. More specifically, for r^{th} order RM code, all coordinates labelled by polynomials of degree greater than r are set to zero. Then the codeword is obtained as binary Reed–Muller transform of this extended data vector [77]. The process of recovering the information bits is then performed by taking the inverse binary Reed–Muller transform of the decoded code vector.

The proposed encoding algorithm holds for any r^{th} order RM code. Moreover, this encoding can be efficiently implemented, as binary Reed–Muller transform can be expressed as Kronecker's direct product of matrices which leads to faster implementation like WHT. We now consider some simple examples to illustrate the encoding procedure.

3.2 A review of TSKN decoding algorithm for RM codes

In Section 2.3 we mentioned that major disadvantage of majority logic approach to decoding RM codes are that as the number of errors increases more stages of majority logic decoding are required, and that the complexity of the majority logic circuitry is exponential. Though these drawbacks have been ameliorated to some extent by sacrificing decoding delay and other strategies in [62]–[63], there are other decoding schemes which lead to lesser decoding delay.

An alternative approach to decoding RM codes is taken by Tokiwa et. al., using the superimposition (concatenation) property of RM codes, to derive decoding algorithms for $R(r, 2^m)$ code [46]. The idea is to decompose larger RM code into several component RM subcodes which can be easily and efficiently decoded. They have given a decoding algorithm for general $R(r, 2^m)$ based on decomposition and also shown that their decoding scheme has significantly less delay compared to conventional majority logic decoding algorithms.

TSKN decoding algorithm holds for any general r^{th} order RM code. At the time of decoding, the process of decomposition, the inverse operation corresponding to superimposition, is employed to obtain the component subcodes. The decomposition is repeated until the stage at which simple iterated (SI) single–error correcting and double–error detecting (SEC–DED) codes are obtained. We next consider decomposition algorithm for obtaining the component subcodes.

3.2.1 Decomposition algorithms for obtaining subcodes

This is best explained with the help of an example. Consider the decomposition of a codeword c of $R(2, 2^6)$ as shown in Figure 3.2.1. To begin with c is decomposed in to $c_1 \in R(1, 2^5)$ and $|c_2|c_2| \in R(2, 2^5)$.
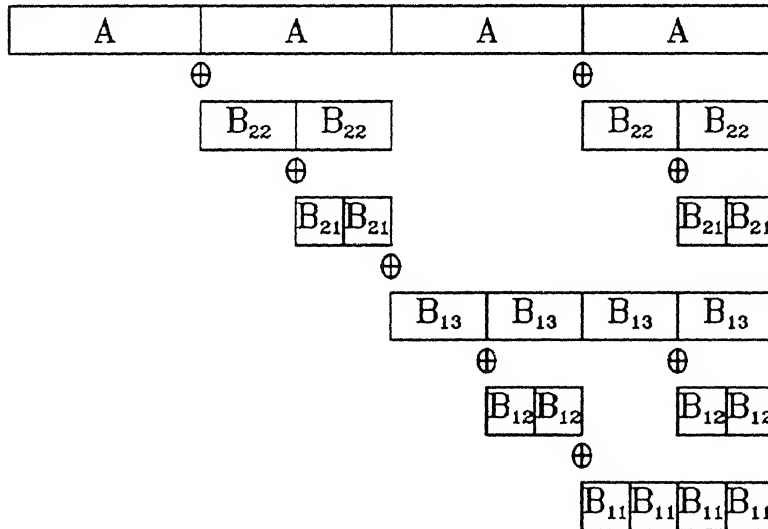
Figure 3.2.1 Decomposition subcodes of $R(2, 2^6)$

The codewords of component $R(2,2^5)$, $R(1,2^5)$ and $R(1,2^4)$ are similarly decomposed. Finally when the decomposition stops we have, starting from the top, 4 copies of A ϵ $R(2,2^4)$; 4 copies of B_{22} and B_{13} belonging to $R(1,2^3)$; 4 copies of B_{11}, B_{12} and B_{21} belonging to $R(0,2^2)$. In general, the following theorem can be proved.

Theorem 3.2.1 : Given an integer v ϵ {1,2,...,m−r} the $(r,2^m)$ code can be decomposed into the $(r-j, 2^{m-v-j})$ RM codes (j = 0, 1,..., r) with the same minimum distance 2^{m-v-j}, each of constitutes the 2^v−SI code.

Choice of v = m−r−2 is of interest. For this choice we have

Theorem 3.2.2 : The $(r,2^m)$ RM code can be decomposed into the 2^{m-r-2}−SI code on the basis of the SEC−DED code of length 2^i and order (i−2) for i = 2, 3, ..., r+2.

3.2.2 TSKN decoding algorithm

Theorem 3.2.2 (v=m−r−2 case) shows that decoding of subcodes consisting of simple iterations of SEC−DED code, constitutes an important subalgorithm. The following algorithm performs decoding of i−SI codes (i denotes the number of times the code is iterated) assuming the decoding algorithm for the component code [46].

Decoding Algorithm for i−SI codes

Let the received word be $|c \oplus e_1|c \oplus e_2|...|c \oplus e_I|$.

1) Set k = 1.

2) Decode $|c\oplus e_k|$ to c_k using any appropriate method.

3) If an error−correction is made in step 2, compute N_k from

$$N_k = \sum_{j=1}^{I} wt(c \oplus e_j \oplus c_k)$$

else, if an error−detection is made in step 2, go to step 5

4) If $N_k < id/2$ go to step 6.

5) If k < i, let k+1 $\longrightarrow$ k and go to step 2.

6) c = c_k and the error−correction is complete

TSKN decoding algorithm reduces the decoding of larger RM codes to decoding the component smaller SEC−DED RM codes which can be efficiently decoded with minimal hardware as discussed in [78]. It is to be noted that the worst case time delay for decoding of repeated i−SI case is 'it$_d$', where t$_d$ is the decoding time for the component code.

The important steps in the TSKN decoding algorithm are summarized as below.

TSKN Decoding Algorithm for $R(r, 2^m)$

1)   Obtain the regenerated vector corresponding to 2^{m-r-2}-SI code.

2)   Decode on the basis of decoding algorithm for i-SI code.

3)   Eliminate the decoded sub-code vector and return to Step-1.

Example 3.2.1 : Consider a code word of $R(2, 2^5)$. The dimension of the code is 16 and minimum distance is 8. So it has an error correcting capability of 3. The decomposition of the code is shown in Figure 3.2.2. Choose the following code word for the transmission purposes. $c = |c_1|c_2|$, where

$$c_1 = 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1$$

$$c_2 = 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1$$

and suppose the error pattern is

$$e_1 = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

$$e_2 = 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0$$

According to Theorem 3.2.1, there are 3 levels of decoding of length 4,8 and 16 respectively. The repetition factor is equal to 2.
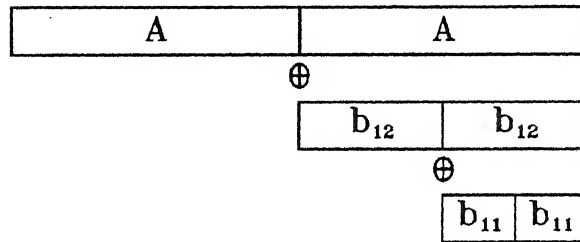
Figure 3.2.2 Decomposition of $R(2, 2^5)$

1)   <u>Decoding of $b_1 \in R(1, 2^5)$</u>

The decoding of b_1 is based on decomposition $b_1 = |b_{12}|b_{12}| \oplus |0|b_1'|$, where $b_1' = |b_{11}|b_{11}|$. The regenerated vector corresponding to b_1, namely r_1, is obtained as

$r_1 = |c_1 \oplus c_2 \oplus e_1 \oplus e_2|$. Then

$$r_1 = 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0$$

1a)   <u>Decoding of $b_1^!$</u>

   $r_1^!$, the regenerated vector corresponding to $b_1^!$ is obtained as follows:

   $$r_1^! = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1$$

Since $b_1^! = |b_{11}|b_{11}|$, b_{11} is to be decoded on the basis of 2–SI $R(0,2^2)$. We have the threshold $id/2 = 4$. It is clear that $b_{11} = 0000$, as the other choice of $b_{11} = 1111$ makes the Hamming weight of $(r_1^! + b_1^!)$ equal to 5 which is greater than threshold value 4.

1b)   <u>Decoding of b_{12}</u>

   In this stage the codeword to be decoded on the basis of 2–SI $R(1,2^3)$. We utilize the dual property of RM codes that $R(1,2^3)$ is self orthogonal [7]. The r_{22}, the regenerated vector corresponding to b_{22} is $|0110\ |\ 1111\ |\ 0111\ |\ 0110\ |$. Computing the parity checks using the v_1, v_2 and v_3 of RM code of length 8, we find that for the first block of r_{22} syndrome is (110) and furthermore the parity check corresponding to v_0 is 0. From this double–error detection condition is identified. Computing the parities of the next half–block of r_{12}, we have syndrome (110) and syndrome w.r.t v_0 1, which indicates an error at position 3 from which $b_{12} = |\ 0110\ |\ 0110\ |$. Therefore b_1 is regenerated as

   $$b_1 = 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0$$

2)   <u>Decoding of A_1</u>

   Now subtracting the contribution of b_1, we have

   $$r_1 = 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1$$

   $$0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1$$

This has to be decoded on the basis of 2–SI $R(2,2^4)$. Since this code is dual to $R(1,2^4)$, we proceed in a similar way as in decoding of $R(1,2^3)$ and obtain A as $|\ 0011\ |\ 0011\ |0011\ |\ 0011\ |$. This completes the decoding.

   TSKN decoding algorithm for RM codes shows that by decomposing into component smaller RM codes and efficiently decoding them, decoding delay is considerably reduced. Note that the process of decomposition, decoding and regeneration of component codes proceeds in a serial fashion. It is therefore interesting to investigate the possibility of alternative way of decomposing a given RM code involving less number of decoding stages compared to TSKN algorithm. Such a decoding algorithm may have smaller delay than TSKN algorithm. The new decoding algorithm may also have reduced memory requirements compared to TSKN algorithm for decoder implementation, as memory required depends on the number of stages in the decoding algorithm.

In view of simple nature of decoding operations in WHT domain as outlined in Section 2.3, we feel that WHT domain spectral techniques are interesting to explore in this regard. In Section 3.4, we first explore the possibilities of WHT domain decoding for RM codes by directly processing WHT of the received vector. Then in subsequent sections we combine superimposition technique of TSKN algorithm to give decoding algorithms for RM codes of order 2 and 3 based on WHT domain decoding subalgorithms. Towards this end, we digress to develop the necessary theoretical tools in the next section.

3.3 A result on the histogram structure of WHT spectrum of $R(2,2^m)$

The choice of mapping from binary alphabet to real number followed is binary $0 \rightarrow 1$ (real) and binary $1 \rightarrow -1$ (real) which is helpful in identifying the histogram structure of the code's WHT spectrum. If a vector over GF(2) is denoted by e then its mapping to the real $\{1,-1\}$ is denoted by $\hat{e}$.

We first prove some lemmas that facilitate expressing binary vector addition and pointwise vector multiplication in terms of corresponding WHT transform domain operations such as addition and convolution.

Lemma 3.3.1 : Let $a = (a_0, a_1, ..., a_{n-1})$ and $b = (b_0, b_1, ..., b_{n-1})$ denote two n–tuples over GF(2). Define $\hat{a}$, corresponding to a as follows. ('$^$' notation is for converting binary to real) $= (1-2a_0, ... , 1-2a_{n-1})$. If $\hat{a} \mapsto \hat{A}$ and $\hat{b} \mapsto \hat{B}$, then

$$a \oplus b \mapsto \hat{a}.\hat{b} \mapsto 1/2^m \, \hat{A} \otimes \hat{B} \quad (n=2^m)$$

Proof: The ith component of binary vector sum $a \oplus b$, when equivalently expressed in terms of arithmetic operation, is translated into pointwise multiplication $\hat{a}_i.\hat{b}_i$. (See Section 2.3). Then the result follows from the convolution property of WHT. □

Lemma 3.3.2 : Let $a = (a_0, a_1, ..., a_{n-1})$ and $b = (b_0, b_1, ..., b_{n-1})$ denote two n–tuples over GF(2) and define pointwise product of a and b as $a.b = (a_0 b_0, ... , a_{n-1} b_{n-1})$. Then

$$(\hat{a.b}) \mapsto 2^{m-1}\delta(i) + \hat{A}_i/2 + \hat{B}_i/2 - \hat{A} \otimes \hat{B}/2^{m+1}$$

where $\delta(i)$ is a 2^m tuple having value 1 at the location $i = 0$ and value 0 elsewhere.

Proof: Equivalent formula for binary '$a_i.b_i$' under '$^$' notation is given by

$$1 - ((1-\hat{a}_i)(1-\hat{b}_i)/2) = 1/2 + \hat{a}_i/2 + \hat{b}_i/2 - \hat{a}_i.\hat{b}_i/2$$

Then lemma follows as a consequence of definition of WHT and Lemma 3.3.1. □

Let v_i represent i^{th} first order Reed–Muller basis vector over binary. Unless otherwise specified, the V, the WHT of the RM code vector v, we mean WHT of the real vector over $\{1,-1\}$ corresponding to v.

Then we have from Section 2.3 that the WHT spectrum of v_i has only one nonzero component with the value 2^m at the position 2^{i-1} (for $i = 1$ to m). The WHT spectrum of v_0 has a nonzero at zeroth location with the value -2^m.

Lemma 3.3.3 : If $v \longmapsto V$, then $v \oplus v_0 \longmapsto -V$.

Proof: Follows from Lemma 3.3.1 and noting from Section 2.3 that $v_0 \longmapsto V_0$ has a single nonzero spectrum at position 0 with a value of -2^m. □

Remark 3.3.1 : From Lemma 3.3.3 we have that the histogram of WHT spectrum of a vector v is is not altered by the addition of v_0. Therefore it follows that to determine the histogram of the WHT spectrum of the RM code, it is enough to consider the histogram of the spectrum of the code subspace spanned by $\{B\} - \{v_0\}$, where $\{B\}$ represents the basis vectors of the generator matrix of the RM code.

The following theorem gives the histogram structure of the generating basis vectors of $R(r,2^m)$.

Theorem 3.3.1: Basis vectors of $R(r,2^m)$ have the following WHT spectrum:

1) The WHT of first order basis vector v_i, for $i=1$ to m, has only one nonzero component, having a value of $\pm 2^m$, at the position 2^i.

2) For $r \geq 2$, the WHT spectrum of the rth order basis vector, $\prod_{k=1}^{r} v_{i_k}$, has 2^r nonzero components; the spectral component at zeroth location has a value $+(2^m - 2^{m-r+1})$ and the remaining $(2^r - 1)$ nonzero components have a value of $\pm 2^{m-r+1}$. The nonzero components of the WHT spectrum are at positions $\sum_{k=1}^{r} a_k 2^{i_k - 1}$, where $a_k = 0$ or 1, for $k = 1$ to r.

Proof: Proof of (1) follows from the definition of basis vectors of first order RM code. For the proof of (2), the method of induction is used. Let $b_s = v_{i_1} v_{i_2} ... v_{i_s}$ be a basis vector of the sth order RM code. Let $b_s \longmapsto B_s$ be WHT pairs. Let $B_s^{(k)}$ denote value of kth component of B_s. Then we have

$$B_s^{(0)} = \pm(2^m - 2^{m-s+1}), \quad B_s^{(r)} = \pm 2^{m-s+1},$$

where $r \in S_s = \{z \in Z \mid z = \sum_{k=1}^{s} a_k 2^{i_k - 1}$, where $a_k = 0$ or 1, for $k = 1$ to $s\}$. Consider a $s+1^{th}$ basis vector $b_{s+1} = b_s v_{i_{s+1}}$. The WHT transform of $v_{i_{s+1}}$ is $V_{i_{s+1}}$ which has a value $+2^m$ at position $2^{i_{s+1} - 1}$. Using Lemma 3.3.2, it follows that

$$B_{s+1} = 2^{m-1} \delta(i) + B_s/2 + V_{i_{s+1}}/2 - B_s \otimes V_{i_{s+1}}/2^{m+1} \tag{3.3.1}$$

Note that although the expression in (3.3.1) contains a convolution term, the result of convolution is simple to analyze because $V_{i_{s+1}}$ has only one nonzero spectral component. The effect of convolution in this case is shift the nonzero spectral components of B_s. The computation of B_{s+1} is divided into four different cases. We start with the zeroth term of B_{s+1}.

a) For computing $B_{s+1}^{(0)}$, $2^{m-1}\delta(i)$ and B_s have nonzero contributions to the sum.

$$B_{s+1}^{(0)} = 2^{m-1} + B_s^{(0)}/2 = 2^{m-1} + (2^{m-1} - 2^{m-(s-1)-1}) = 2^m - 2^{m-s}$$

b) For the case of $r \in S_s$, only $B_s^{(r)}/2$ is nonzero and the other terms are zero valued. Therefore $B_{s+1}^{(r)} = \pm 2^{m-s}$.

c) For the computation of i_{s+1}^{th} term of B_{s+1}, $V_{i_{s+1}}$ and the i_{s+1}^{th} term of convolution $B_s \otimes V_{i_{s+1}}/2^{m+1}$ participate. Therefore

$$B_{s+1}^{(i_{s+1})} = [V_{i_{s+1}}/2] - [(2^m - 2^{m-s+1})2^m/2^{m+1}] = 2^{m-1} - 2^{m-1} + 2^{m-s} = 2^{m-s}$$

d) Note that for each $r \in S_s$, $i_{s+1} \oplus r$ are all different. Hence $B_s \otimes V_{i_{s+1}}/2^{m+1}$ term contributes nonzero value at positions belonging to to $S_{s+1} = \{i_{s+1} \oplus t \mid t \in S_s\}$. Therefore for $t \in S_{s+1}$ and $t \neq i_{s+1}$, we have

$$B_{s+1}^{(t)} = (\pm 2^{m-s+1}2^m/2^{m+1}) = \pm 2^{m-s}$$

This completes the proof of the theorem. □

Example 3.3.1 : Consider the codewords of $R(2,2^4)$. There are $\binom{4}{1} + \binom{4}{2}$ basis vectors in the generator matrix of the code. The histogram for $R(2,2^4)$ is given in Table 3.3.1.

From now on, we shall concentrate exclusively on $R(2,2^m)$ codes. We prove two lemmas that establish the histogram structure for certain codewords of $R(2,2^m)$ having a particular pattern. It will be shown later that these lemmas together contain all the histogram classes in the WHT spectrum of $R(2,2^m)$.

Lemma 3.3.4 : Walsh–Hadamard spectrum of $v_i v_j \oplus v_s v_t$, $i \neq j \neq s \neq t$, has 16 nonzero positions. Furthermore each nonzero position has value $\pm 2^{m-2}$.

Proof: We know from Theorem 3.3.1 that WHT spectrum of $v_i v_j$ has nonzeros at positions at $0, 2^{i-1}, 2^{j-1}$ and $2^{i-1} + 2^{j-1}$. Similarly $v_s v_t$ has nonzeros at positions 0, 2^{s-1}, 2^{t-1}, $2^{s-1} + 2^{t-1}$. We have from Lemma 3.3.1 that $(v_i v_j \oplus v_s v_t) \longmapsto (1/2^m) V_{ij} \otimes V_{st}$, where V_{ij} is the WHT of $v_i v_j$. Because $i \neq j \neq k \neq l$, each of the

Table 3.3.1 Histogram of WHT of basis codewords of R(2,2^4) code

Basis vectors		Non—zero Spectrum	Position(s)
v_1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	16	1
v_2	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1	16	2
v_3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1	16	4
v_4	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	16	8
v_1v_2	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1	8,8,8,−8	(0,1,2,3)
v_1v_3	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1	8,8,8,−8	(0,1,4,5)
v_1v_4	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1	8,8,8,−8	(0,1,8,9)
v_2v_3	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1	8,8,8,−8	(0,2,4,6)
v_2v_4	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1	8,8,8,−8	(0,2,8,10)
v_3v_4	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	8,8,8,−8	(0,4,8,12)

nonzero spectral components of V_{st} multiplies all the nonzero spectral components of V_{ij} to give rise to different set of nonzero positions in the final convolution. It is easily seen that the values of each of the nonzero spectral component is $\pm(2^{(2m-2)}/2^m) = \pm2^{m-2}$. $\quad\square$

The above lemma is easily generalized.

Lemma 3.3.5: Walsh_Hadamard transform of $v_{i_1}v_{i_2} \oplus ... \oplus v_{i_{2k-1}}v_{i_{2k}}$, where all i_k's are distinct, has 2^{2k} nonzero positions each nonzero position having a value of $\pm2^{m-k}$.

Proof: Let us proceed by induction. Suppose the lemma is true for n=k−1. Then we have 2^{2k-2} nonzeros each having a value of $\pm2^{m-k+1}$. After '$v_{2k-1}v_{2k}$' is added, as 2k $\neq$ 2k−1, the number of nonzero factors increase by a factor of 4. And further value of each nonzero component has a value $\pm(2^{m-k+1}x2^{m-1}/2^m)$ $=\pm2^{m-k}$. $\quad\square$

In general, when there are more than two summands in a second degree expression involving v_i's, it is difficult to analyze the histogram structure by first principles. Only in special cases, as in Lemmas 3.3.4 and 3.3.5, that we can prove some results. One way out of this situation is to find some transformation of variables that may reduce any given second degree expression to some standard form whose WHT spectrum reduces to the cases of Lemma 3.3.4 and 3.3.5. That the transformed vector also belongs to the

code follows from automorphism property of Reed–Muller codes. Fortunately, at least for second order RM codes, Dickson's Theorem (see Theorem 3.3.2) is available which precisely does this job. A proof of the theorem is given in [7]. We later invoke this important theorem in obtaining the histogram structure of WHT spectrum of $R(2,2^m)$.

We need some notations and terminology before stating Dickson's Theorem, The following various ways of representing codewords of second order RM codes are useful. A typical codeword of a $R(2,2^m)$ is represented in terms of Boolean polynomials as

$$S(x) = \sum_{1 \le i, j \le m} q_{ij} x_i x_j \oplus \sum_{1 \le i \le m} l_i x_i \oplus \epsilon$$

This can be alternatively written as

$$S(x) = x\, Q\, x^T \oplus L\, x^T \oplus \epsilon,$$

where $Q = (q_{i,j})$ is an upper triangular matrix, $L = (l_1, l_2, ..., l_m)$ is a binary vector, and ϵ is 0 or 1. Here Q is referred to as quadratic form and L is referred to as linear form.

Example 3.3.2 : Consider a codeword of $R(2,2^4)$ which has the following polynomial form '$v_1 v_2 \oplus v_3 v_4 \oplus v_1 \oplus 1$'. Then we have

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \qquad Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and $L = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ and $\epsilon = 1$.

Theorem 3.3.2 (Dickson): (1) Any Boolean function of degree ≤ 2,

$$xQx^T \oplus L(x) \oplus \epsilon$$

where Q is an upper triangular matrix and L, ϵ are arbitrary, becomes

$$T(y) = \sum_{i=1}^{h} y_{2i-1} y_{2i} + L_1(y) + \epsilon$$

under the transformation of variables $y = xR^{-1}$. (For details of how to obtain R from Q refer to [7]) Moreover $y_1, ..., y_{2h}$ are linearly independent.

(2) If $L_1(y)$ is linearly dependent on $y_1, ..., y_{2h}$, we may by an affine transformation of variables, write $T(y)$ as

$$\sum_{i=1}^{h} z_{2i-1} z_{2i} \oplus \epsilon_1, \quad \epsilon_1 \text{ is 0 or 1}$$

where $z_1, ..., z_{2h}$ are linearly independent, and each z_i is a linear form in $y_1, ..., y_{2h}$, and 1.

Just as we require Dickson's theorem to simplify time domain description of a codeword of $R(2,2^m)$, the corresponding changes in the WHT domain is important for our purposes. The following lemma shows how this connection obtains, in general, for any real vector a and in particular for our case.

Lemma 3.3.6 : If $a \longmapsto A$ and $b \longmapsto B$ are two WHT pairs, and if $b = P(T)a$, where T is an affine transformation and $P(T)$ is the corresponding permutation matrix, then there exists an affine transformation T^T such that $B = P(T^T)A$.

Proof: Let $P(T)$ denote the permutation matrix corresponding to an affine transformation T. Then $b = P(T)a$ denotes the transformation of the vector a over reals. If H denotes the WHT matrix, then

$$B = Hb = HP(T)a$$

It is known that the $(i,j)^{th}$ of H (WHT matrix) is given by $(-1)^{i.j}$, where $i.j = (i_0 j_0 \oplus i_1 j_1 \oplus ... \oplus i_{m-1} j_{m-1})$. We see that $P(T)$ permutes the column of H. The affine transformation T modifies the j_k^{th} bit to $j_k' = (j_0 t_{k,0} \oplus j_1 t_{k,1} \oplus ... \oplus j_{m-1} t_{k,m-1})$ for k=0 to m−1, where $t_{i,j}$ denotes the $(i,j)^{th}$ element of T. Accordingly i.j is transformed to $(i_0 j_0' \oplus i_1 j_1' \oplus ... \oplus i_{(m-1)} j_{(m-1)}')$. Rearranging this summation as $(i_0' j_0 \oplus i_1' j_1 \oplus ... \oplus i_{(m-1)}' j_{(m-1)})$, it is clear that $i_k' = (i_0 t_{0,k} \oplus i_1 t_{1,k} \oplus ... \oplus i_{m-1} t_{m-1,k})$, for k=0 to m−1. This affine transformation is easily seen to be T^T, where superscript refers to transpose. In other words, we have proved that $HP(T) = P(T^T)H$. Therefore

$$B = HP(T)a = P(T^T)Ha = P(T^T)A \qquad\qquad \Box$$

Remark 3.3.2 : The above lemma has the following important implication. This lemma, in effect, says that the histogram structure of the WHT of a sample domain vector is invariant under a given affine permutation of the sample domain vector.

Example 3.3.3 : Consider the following codeword given by Boolean expression $"x_1(x_2 \oplus x_3) \oplus x_3 x_4"$ belonging to $R(2,2^4)$. Then the code vector f is

        0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1

Its transform F is given by

        4 4 4 −4 4 4 4 −4 4 4 −4 4 −4 −4 4 −4

Consider the affine transformation $x_1' \rightarrow x_1$, $x_2' \rightarrow x_2 \oplus x_3$, $x_3' \rightarrow x_3$ and $x_4' \rightarrow x_4$. The matrix T and therefore T^T are given by

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Under the transformation T the boolean polynomial changes to "$x_1' x_2' \oplus x_3' x_4'$". In this case the WHT of the permuted vector $F' = P(T)f$ is as follows

$$4\ 4\ 4\ -4\ 4\ 4\ 4\ -4\ 4\ 4\ 4\ -4\ -4\ -4\ -4\ 4$$

The row permutation effecting the same change is given by the affine transformation $x_1' \to x_1$, $x_2' \to x_2$, $x_3' \to x_2 \oplus x_3$ and $x_4' \to x_4$, corresponding to T^T. It is easily verified that F' is obtained from F by interchanging the coordinates 2 and 3 with 6 and 7 as well as the coordinates 10 and 11 with the coordinates 14 and 15, respectively.

We now state and prove the following important theorem of this section on the structure of WHT coefficients of $R(2,2^m)$.

Theorem 3.3.3: The WHT spectrum of $R(2,2^m)$ codewords can be classified into $\lfloor m/2 \rfloor + 1$ different classes. The structure of WHT belonging to class j, for j=0 to $\lfloor m/2 \rfloor$, has 2^{2j} nonzero spectral values, each nonzero value being equal to $\pm 2^{m-j}$.

Proof: The codewords of $R(2,2^m)$ can be expressed as boolean polynomials of degree ≤ 2. The first degree polynomials contribute a histogram class having a single nonzero with value $\pm 2^m$. For a second degree polynomial, by Dickson's theorem there exists a linear transformation which carries the codeword into canonical form $(\sum_{i,j} y_i y_j \oplus l(y) \oplus \epsilon)$. First consider the case $l(y) = 0$.

a) The canonical structure is of the form '$y_1 y_2 \oplus \ldots \oplus y_{2k-1} y_{2k}$'. If m is even, then k ranges from 1 to p, where m=2p; If m is odd, then k ranges from 1 to p where m=2p−1. Therefore there are p = $\lfloor m/2 \rfloor$ different canonical classes corresponding to p canonical forms. From Lemma 3.3.5, it follows that the jth canonical class has 2^{2j} nonzero spectral values, each equal to $\pm 2^{m-j}$. Lemma 3.3.6 guarantees that the histogram class of any time domain vector is the same as the histogram class of its corresponding canonical vector.

b) If $l(y)$ is not zero, Let F represent the WHT of the quadratic term and and G represent the WHT corresponding to $l(y)$. Since L has only one nonzero spectral value, it follows from Lemma 3.3.1 that the histogram structure of WHT of $F \circledast G$ is the same as the histogram structure of F. $\square$

Table 3.3.2 gives histogram structures for some second order RM codes. The entries in the "frequency" column represent the number of nonzero spectral components in a histogram class having same value.

Table 3.3.2: Histograms of WHT spectrum of some second order RM Codes

Frequency	Value
1	±16
4	±8
16	±4

$R(2,2^4)$

Frequency	Value
1	±32
4	±16
16	±8

$R(2,2^5)$

Frequency	Value
1	±64
4	±32
16	±16
64	±8

$R(2,2^6)$

Frequency	Value
1	±128
4	±64
16	±32
64	±16

$R(2,2^7)$

We shall need the following lemma later on. It is concerned with proving more details of the histogram structure of WHT of $R(2,2^4)$.

Lemma 3.3.7: The WHT of $R(2,2^4)$ code has the following pattern of positive and negative spectral values:

1) The class of codewords whose WHT spectrum has a single nonzero component of value ±16.

2) The class of codewords whose WHT spectrum has four 8's have either 3 positive 8's or 3 negative 8's.

3) The class of codewords whose WHT spectrum has sixteen ±4's have either 10 positive 4's or 10 −ve 4's.

Proof: By virtue of Theorem 3.3.3, $R(2,2^4)$ has three distinct classes of WHT spectrum corresponding to v_i, $v_i v_j$, $v_i v_j \oplus v_k v_t$ respectively. By Lemma 3.3.3, if v_0 is added to any codeword v independent of v_0, then the sign of components of V, the WHT of v, is reversed and therefore it is enough to consider v. The first of these classes correspond to codewords of $R(1,2^4)$ whose spectrum has a single nonzero with absolute value 16. For the second class of codewords by Lemma 3.3.2 it is immediate that for the codeword $v_i v_j$, the spectrum at the position $2^{j-1}+2^{j-1}$ is negative valued, whereas the other nonzero spectrum are positive valued. It then follows for the codewords of the form $v_i v_j \oplus v_k v_t$, each of 3 positive valued spectrum of $v_i v_j$ combined with 3 positive valued spectrum of $v_k v_t$ contribute 9 positive spectral values and two negative valued spectrum together contribute one more positive valued spectrum. This completes the proof. □

Remark 3.3.3: Unfortunately, results on the histogram structure of WHT could not be extended for RM codes of order ≥ 3. This is because the affine transformations in these cases are not available to reduce Boolean expressions of degree ≥ 3 to some canonical form. Some studies on determining the weight structure of RM codes do require such theorems concerning canonical forms. However, only partial results are available in the literature [7]. In particular, we refer to a theorem proved by Kasami [Theorem 10, [7]]. The main limitation of Kasami's theorem is that there is a restriction on the weight of the codeword whose canonical simplification is to be found. Any breakthrough in this direction, no doubt, will be useful for deriving the histogram structure of Reed–Muller codes of order > 2.

3.4 WHT domain decoding of $R(2,2^4)$

We begin by explaining the reasons for not considering direct decoding $R(2,2^m)$ upto $[d_{min} -1/2]$ by WHT techniques. First of all, let us check whether the WHT spectrum of $R(2,2^m)$ satisfies two necessary conditions for recoverability of histogram pattern mentioned in Section 2.3.3. From Theorem 3.3.3, the minimum spectral value in the WHT spectrum of $R(2,2^m)$ is $2^{\lceil m/2 \rceil}$. As $R(2,2^m)$ has the minimum distance 2^{m-2}, it can correct upto $(2^{m-3}-1)$ errors. We have also shown that an e–random error alters the value of the codeword spectrum by at most 2e. Therefore for the case of $R(2,2^m)$, the maximum possible deviation in a spectral component is $(2^{m-2}-2)$. As explained in Section 2.3.3, the first condition to be satisfied is that modified nonzero spectral component should not change its sign. This implies that 'm' should be so chosen that $2^{m-2}-2 \leq 2^{\lceil m/2 \rceil}$, or m $\leq$ 4. This means that direct WHT method is not possible for decoding of $R(2,2^m)$. The case m=4 corresponds to $R(2,2^4)$, which is the first nontrivial code of length 16 having single–error correction and double–error detection capabilities.

We first give a new WHT domain algorithm to decode $R(2,2^4)$ for correcting single errors. This algorithm is followed by a special WHT based decoding algorithm for decoding $2^{m-4}*R(2,2^4)$ (iterated code), one of the main subalgorithms in our forthcoming decoding algorithm for $R(2,2^m)$. This section also intends to give a flavor of WHT based analytical techniques while attempting to give a semi–rigorous proof of the WHT decoding algorithm.

In order to develop WHT based algorithm for $R(2,2^4)$, we have to know the modified histogram structure of $R(2,2^4)$ when a single error is added to any codeword. Our technique of determining the modified spectrum by analyzing the effect of an error on a codeword spectrum is as follows. Since the received received vector is expressed in ±1, an error at any particular position may change −1 to 1 or 1 to −1. This can be thought of as sum of transmitted code vector and an auxiliary vector having nonzero value at the position of error with the value ±2. Therefore the WHT spectrum of the received vector is obtained by adding to the transmitted codeword spectrum the column of WHT matrix corresponding to the error location multiplied by ±2.

The histogram analysis mentioned in the above paragraph is, in general, a difficult problem to solve analytically because it involves some of properties of Walsh–Hadamard matrix in a deeper way. Exhaustive computer search methods are not possible except for small error cases. However, as the code $R(2,2^4)$ is of smaller length, it is feasible to do exhaustive computer search and the results are presented below.

Theorem 3.4.1 : Consider a coset of $R(2,2^4)$ code formed when a single error is added to the code. Then the histogram structure of the coset is as given below in Table 3.4.1.

Table 3.4.1 Modified histogram of $R(2,2^4)$ due to addition of a single error

Pattern No.	Spectrum values	Histogram
1	2,14	15,1
2	2,6,10	12,4
3	2,6	10,6

Proof: There are three cases to be considered, each corresponding to three different histogram classes of $R(2,2^4)$.

(A)   Consider a transmitted codeword whose WHT spectrum has a single nonzero spectrum with the value 16. It is clear that then the codeword belongs to $R(1,2^4)$. Suppose a single error occurs at position i during transmission. Then it is easy to see using the auxiliary vector technique that positions where spectrum values are 0 may change to ±2. As for highest spectral value 16, it can change to either 14 or 18. We want to prove by analyzing the corresponding time domain vector that modification to 18 is not possible. By association of $L(v)$, not containing v_0, with the binary function $L(x)$, where L is a linear function corresponding to the transmitted codeword, the following correspondences are established. If the highest value spectral position is h, then the binary function takes the form $L(x) = h.x$, and the value of time domain vector at any position i is $(-1)^{h.i}$. Let us compute the value of modified term to be added to code spectrum at the position h, if an error happens at position i. If the value of code word at i^{th} location is given by $(-1)^{h.i}$, then the nonzero component of the auxiliary vector has a value $-2(-1)^{h.i}$. The factor '−2' arises because an error changes code bit +1 to −1 and vice versa. The modified spectrum is obtained by adding i^{th} column of WHT matrix to codeword spectrum. It is known that $(i,j)^{th}$ element of WHT matrix is $(-1)^{i.j}$. Therefore the maximum valued location h is modified by the addition of $-2(-1)^{h.i}(-1)^{h.i}$ or −2, and therefore the highest spectral value 16 goes to 14. If $L(v)$ contains v_0, note that the binary polynomial expression is given by $L(x) \oplus 1$. Therefore both the code spectrum and the modifying factor are multiplied by −1. This means that highest code spectral value is modified to −14.

(B)   Considering the second entry in the Table 3.4.1, it is clear that the WHT spectrum of the codeword transmitted has four 8's. It is known from the code's histogram structure that of the four 8's 3 are either $+8$ or -8 (The latter corresponds to the case when v_0 is added to it). It is also known from Theorem 3.3.1 and Lemma 3.3.1 that the location of these nonzero values are at positions $h, h \oplus i, h \oplus j, h \oplus i \oplus j$, for some vector h. Suppose $v_i v_j \oplus L(v)$ be the time domain vector not containing v_0. The corresponding binary polynomial is $x_i x_j \oplus L(x)$, where L is linear function of x's. Suppose that the maximum spectral component of $L(x)$ is at position 'h', then at any position 'k, the value of code vector is $k_i k_j \oplus k.h$. If the error position is at 'k', then the spectral value at any position 'm' changes by $-2(-1)^{k_i k_j \oplus k.h \oplus k.m}$. Therefore for observation positions $h, h \oplus i, h \oplus j, h \oplus i \oplus j$, the respective changes in the spectral values are A, $A(-1)^{k.i}$, $A(-1)^{k.j}$ and $A(-1)^{k.(i \oplus j)}$, where $A = -2(-1)^{k_i k_j}$. This can be further simplified to A, $A(-1)^{k_i}$, $A(-1)^{k_j}$ and $A(-1)^{k_i \oplus k_j}$, because i and j are the maximal spectral location corresponding to first order basis vectors v_i and v_j. Different possibilities for modifying factors are $(-2,-2,-2,-2)$ for $k_i = k_j = 0$, $(-2,2,-2,2)$ for $k_i = 1$ and $k_j = 0$, $(-2,-2,2,2)$ for $k_i = 0$ and $k_j = 1$, and $(2,-2,-2,2)$ for $k_i = k_j = 1$. Since the spectral value pattern corresponding to these positions are $(8,8,8,-8)$ the desired result follows.

    If the codeword contains v_0, note that the binary polynomial expression is given by "$x_i x_j \oplus L(x) \oplus 1$". Then all the code spectrum values will be multiplied by -1 and likewise the expressions for modification factors are also multiplied by -1. Therefore the above result is true for this case also.

(C)   Suppose $v_i v_j \oplus v_g v_h$ be the code vector and the corresponding polynomial is $x_i x_j \oplus x_g x_h$. Suppose an error occurs at position 'e'. Then the value of code vector at position e is $(-1)^{e_i e_j \oplus e_g e_h}$. Therefore the spectral change at r^{th} position is given by $(-2)(-1)^{e_i e_j \oplus e_h e_g \oplus r.e}$. An important property of the codeword we now invoke is that the spectrum of the codeword is a multiple (4 times in this case) of the code vector [7]. Therefore the net change in r^{th} spectral component is given

$$[4(-1)^{r_i r_j \oplus r_g r_h} - 2(-1)^{e_i e_j \oplus e_h e_g \oplus r.e}]$$

This can be further simplified as

$$2(-1)^{r_i r_j \oplus r_g r_h}[2 - (-1)^{e_i e_j \oplus e_h e_g \oplus r.e \oplus r_i r_j \oplus r_g r_h}] \tag{3.4.1}$$

If the expression within the [ ] brackets of (3.4.1) takes value 1, then the modified spectral value is ± 2, else it is ± 6. The exponent of (-1) within square brackets can be put in a factored form as

$$(e_i \oplus r_j)(r_i \oplus e_j) \oplus (e_h \oplus r_g)(r_h \oplus e_g) \tag{3.4.2}$$

Note that e is fixed and r is varying. It is easy to check that $\bar{e}_i \bar{e}_j e_g e_h$, $\bar{e}_i \bar{e}_j \bar{e}_g e_h$, $\bar{e}_i \bar{e}_j e_g \bar{e}_h$, $e_i \bar{e}_j \bar{e}_g \bar{e}_h$, $\bar{e}_i e_j \bar{e}_g \bar{e}_h$ and

$e_i \bar{e}_j \bar{e}_g \bar{e}_h$ are the only 6 combinations that make value of (3.4.2) −1 and therefore, value of (3.5.1) ±6.

Consider the codeword whose boolean expression is $x_i x_j \oplus x_g x_h \oplus h.x$ and the WHT is F'. Let F be the WHT corresponding to code vector $v_i v_j \oplus v_g v_h$. Then,

$$F'_j = \sum_{k=0}^{n-1} (f_k (-1)^{k.h}) (-1)^{k.j} = \sum_{k=0}^{n-1} f_k (-1)^{k.(j \oplus h)} = F_{j \oplus h}$$

Since F' is a shifted version of F, both F and F' belong to the same histogram class.

Finally, the histogram class of the WHT spectrum of any codeword, by Dickson's theorem, is reducible to to one of the three canonical cases discussed above. This completes the proof of the theorem. □

WHT Domain Decoding Algorithm for $R(2,2^4)$

1) Compute WHT R of the received vector $r = c \oplus e$, mapped to reals.

2) Compute ind_2, the number of components of R whose absolute value is ≤ 2; and ind_6, the number of components of R whose absolute values is >2 but ≤ 6.

3) If ind_2 = 15, then find the maximum absolute valued location and replace the corresponding component of R by +16 or −16 preserving the sign. Replace the rest of the components by zero. Go to step 7.

4) If ind_2 = 12, then replace the positions in R with the absolute value 6 or 10 by 8, preserving the sign. Replace the other spectral components by zero. Go to step 7.

5) If ind_2 = 10 and ind_6 is such that ind_2+ind_6 = 16, then replace +ve spectrum of R by +4 and the −ve spectrum of R by −4. Go to step 7.

6) An uncorrectable pattern is detected. Set uncorrect flag true.

7) If uncorrect flag is false then compute IWHT on R to obtain the transmitted code vector. This may be further processed to obtain the information bits. Else, if the uncorrect flag is true then output the message that the received vector contains uncorrectable errors.

Decoding of $2^{m-4} * R(2,2^4)$

Decoding of simple iteration code $R(2,2^4)$, repeated 2^{m-4} times, is an important subalgorithm of the forthcoming decoding algorithm for $R(2,2^m)$. The process of the decoding as before uses comparison with appropriately chosen thresholds and appropriate substitution operations. We start with a lemma concerning the WHT of a code vector repeated 2^{m-4} times.

Lemma 3.4.1: Let C be the WHT of $c \in R(2,2^4)$. Then the WHT of 2^{m-4}–iterated $|c|c|...|c|$ is given by $|C'|0|0|...|0|$ (0 is repeated in 2^{m-4}−1 blocks of length 16), where $C' = 2^{m-4}C$.

Proof: First consider computing WHT of a 2 times iterated vector $|a|a|$ of length 2^m, where a is of length

2^{m-1}. Invoking direct product property of Hadamard matrix we have,

$$\begin{bmatrix} 2H_{m-1} \cdot a \\ 0 \end{bmatrix} = \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix}$$

The required result follows from repeated application of the above property. $\square$

Remark 3.4.1 : The structure of the codeword suggests that only first 16 components of R need to be processed for decoding. Computation of R proceeds in the usual manner until the final stage of butterfly algorithm and in the final stage compute only first 16 terms. Another elegant solution is to compute WHT of each of 2^{m-4} sub blocks separately and then add the transforms vectorially to get the first 16 terms. It may be noted that parallelism can be employed for computation of WHT of sub-blocks.

We have seen that for the single error case, the modifications of the received spectrum analyzed by the technique of introducing an auxiliary vector having value ±2 at the position of error. This approach can be generalized, in a simple manner, as follows for the case of multiple errors. Consider a random error pattern having weight e_h; divide the error pattern into sub-blocks of length 16 and consider the position of error locations within each sub-block; let e_1 errors occur in the first position of the sub-block, e_2 in the second position of the sub-block and so on. Therefore the auxiliary vector in this case is easily seen to be $(\pm 2e_1, \pm 2e_2, ..., \pm 2e_{16})$, where $e_1 + e_2 + ... + e_{16} = e_h$. And the WHT spectrum of the received vector is obtained by adding to the transmitted codeword spectrum the columns of WHT matrix weighted by corresponding auxiliary vector component.

Unlike the single error case, the histogram classification of modified WHT spectrum for the multiple errors case is not useful in obtaining the decoding algorithm for $2^{m-4}*R(2,2^4)$. This is because the number of histogram classes for the multiple error case are very large compared to single error case and grows monotonically with 'm'. For example, $R(2,2^6)$ has 4 histogram classes in the code spectrum and 220 histogram classes in the modified codeword spectrum due to error! Therefore a different approach to identify the histogram class of a received signal from the knowledge of variations of spectral values is needed.

From Lemma 3.4.1 it follows that the number of histogram classes in the WHT spectrum of $2^{m-4}*R(2,2^4)$ is 3. Each of three classes correspond to the 3 classes in $R(2,2^4)$ with the nonzero spectral values multiplied by a scalar factor of 2^{m-4}. The $d_{min} = 2^{m-2}$ and $e = (2^{m-3}-1)$. Table 3.4.2 shows the respective spectral values and their maximum and minimum range of variation for the case of multiple errors.

The first observation is that spectral values differing by 8 have distinct range of variation. The implication of this observation is that two codewords belonging to the same histogram class are distinguishable from each other as different spectral values differ at least by 8. The second observation is that the range of variations for the spectral value 2^m is easily distinguished from others. Therefore there is no problem identifying the histogram class of the received vector corresponding to a code vector having WHT spectrum consisting of only one nonzero component with value $\pm 2^m$.

Table 3.4.2 Actual spectral values and their range of variation due to errors

Minimum Possible Value	Actual Spectral Value	Maximum Possible Value
$-(2^{m-2}-2)$	0	$(2^{m-2}-2)$
2	2^{m-2} (4)	$(2^{m-1}-2)$
$(2^{m-2}+2)$	2^{m-1} (8)	$(3.2^{m-2}-2)$
$(3.2^{m-2}+2)$	2^m (16)	2^m

(.) denotes the number of times a nonzero spectral values appear.

Lemma 3.4.2 : The range of variation for the codeword whose spectrum has a single nonzero value $\pm 2^m$ is from $3.2^{m-2}+2$ to 2^m.

Proof: It is clear that in this case codeword transmitted belongs to $R(1,2^m)$ subcode of $R(2,2^m)$. Suppose $L(x) = h.x$ be the corresponding binary function. Let an error pattern e be such that the corresponding sub block of length 16 has errors at places $p_1, p_2, ..., p_r$ with multiplicity $e_1, e_2, ..., e_r$, respectively, satisfying $e_1+e_2+..e_r \leq 2^{m-3}-1$. Then if 'r' is the observation position, we have

$$S_r^m = S_r^o - \sum_{k=1}^{r} (-2e_k)\, (-1)^{p_k.h}\, (-1)^{p_k.r}$$

where superscripts 'm' and 't' refer to transmitted and modified codeword spectrum respectively. If $r = h$, then $S_r^o = 2^m$, and the above equation simplifies to $(2^m-2e_1-2e_2...-2e_r)$. This proves the lemma. □

Let us now consider the cases illustrating how ambiguities arise in determining the histogram class of a received signal. It is evident from the Table 3.4.2 that the range of variation for spectral value '0' overlaps with the range of variation for spectral value '2^{m-2}'; that the range of variation for spectral value '2^{m-2}' overlaps with, both the range of variations for spectral value '0' and spectral value '2^{m-1}'. However, as shown above, the range of variation for '2^{m-1}' does not overlap with that of spectral value '2^m'. As a consequence of these overlappings there are cases when the received spectrum can be considered as modification of either the spectrum of the transmitted codeword or the spectrum of another codeword

belonging to different histogram class. For example, suppose transmitted codeword spectrum has 4 nonzero spectrum with the absolute value 2^{m-1}. Suppose that all of the four positions in the codeword spectrum having absolute value 2^{m-1} are downward modified due to channel error. Alternatively they could also be thought of as upward modifications of the spectral value 2^{m-2}. Since the range of variation of spectral values '0' and '2^{m-2}' overlap, the received signal spectrum could be either due to modification of codeword spectrum having 4 nonzero spectrum (value 2^{m-1}) or due to modification of codeword spectrum having 16 nonzero spectrum (value 2^{m-2}). So there is an ambiguity in determining the histogram class of the transmitted signal. The following lemma provides a more deeper insight into nature of this ambiguity.

Lemma 3.4.3: Consider a codeword whose spectrum has 4 nonzero positions with the absolute value 2^{m-1}. Then there are errors which downward modify all the four nonzero components of the spectrum.

Proof. Consider the codeword whose binary polynomial is given by $x_i x_j \oplus x.h$. Then it follows that the nonzero occur in positions h, $h \oplus i$, $h \oplus j$ and $h \oplus i \oplus j$. Consider an error e such that the corresponding 16 length sub-block has errors at places $p_1, p_2, ..., p_t$ with multiplicity $e_1, e_2, ..., e_t$ respectively. Let r be the observation point. S_r denote spectral value at position r.

$$S_r^m = S_r^0 - 2 \sum_{k=1}^{t} e_k (-1)^{(p_k)_{ij}} (-1)^{p_k.(h \oplus r)}$$

where $(p_k)_{ij}$ denotes multiplying i and j bits of p_k. There are only four possible choices for i and j bits are 00, 01, 10, 11 and, correspondingly, the set of p_k's can be subdivided into four subsets P_0, P_1, P_2 and P_3. Define $S_i = \sum_{k \in P_i} e_k$, for $i = 0$ to 3. Note that all S_i's are nonnegative and include contributions from all p_k's having the same i^{th} and j^{th} bits. Further, if we restrict attention to four observation positions h, $h \oplus i$, $h \oplus j$, $h \oplus i \oplus j$, we have

$$S_h^m = S_h^o - 2(S_0 + S_1 + S_2 - S_3)$$
$$S_{h \oplus i}^m = S_{h \oplus i}^o - 2(S_0 + S_1 - S_2 + S_3)$$
$$S_{h \oplus j}^m = S_{h \oplus j}^o - 2(S_0 - S_1 + S_2 + S_3)$$
$$S_{h \oplus i \oplus j}^m = S_{h \oplus i \oplus j}^o - 2(S_0 - S_1 - S_2 - S_3)$$

Since S_i's depend upon the error positions, it is possible to choose an error pattern such that the inequalities $S_1 + S_2 + S_3 > S_0$, $S_0 + S_2 + S_3 > S_1$, $S_0 + S_1 + S_3 > S_2$ and $S_0 + S_1 + S_2 > S_3$ are satisfied. For example the choice of $S_0 = S_1 = S_2 = S_3$ satisfies required conditions. Then it is easy to see that all spectral coefficients are downward modified in absolute value. □

Since there are only 3 different histogram classes and one of the histogram class is unambiguously distinguished, it turns out that, apart from the case discussed above, there are no other ambiguities in identifying the histogram class of the transmitted codeword from the received data. One way of resolving ambiguity is by obtaining complete information on modified histogram structure. But this is not be practicable because of two reasons; first, the number of histogram classes increase with 'm' and second, to obtain all the relevant details of histogram structure may be computationally expensive. An alternative approach is based on the fact that ambiguity is restricted to two cases. This suggests a feasible method of obtaining two possible estimates of the codeword in the time domain based on two possible substitutions in the spectral domain, and decide on the basis of maximum likelihood principle. We follow the latter method in this thesis.

So the overall picture of decoding process that emerges is the following. Select 3 thresholds, each corresponding to maximum possible variation of spectral values 0, 2^{m-2} (4) and 2^{m-1} (8). Note that these threshold values are in the natural ascending order. Count the number of spectral values in the received spectrum that are less than or equal to threshold_1, that are in between thresholds and that are greater than threshold_3. If the histogram class of a received vector is unambiguously identified, then algorithm proceeds to perform appropriate substitutions followed by inverse IWHT to get the code vector. If there is ambiguity in identifying the histogram class, this will be resolved as explained above. The algorithm is presented below.

WHT Domain Decoding Algorithm for $2^{m-4}*R(2,2^4)$

(Algorithm requires three thresholds corresponding to maximum deviation of actual spectral values. First threshold is set at $2^{m-2}-2$ (max_0), the second at $2^{m-1}-2$ (max_4)and the third at $3.2^{m-2}-2$(max_8))

1)   Compute first 16 components of R, the WHT of the real vector corresponding to $r = c \oplus e$.

2)   Perform magnitude comparison on the first 16 components of R with 3 thresholds and find the following:

     1) cnt_1 : Number of components $\leq$ max_0.

     2) cnt_2 : Number of components > max_0 and $\leq$ max_4.

     3) cnt_3 : Number of components > max_2 and $\leq$ max_8.

     4) cnt_4 : Number of components > max_8

3)   If (cnt_1 = 15) and (cnt_4 = 1), then find the maximum absolute valued location of R and replace it by 2^m preserving the sign. Replace the rest of the components with zero. Perform IWHT to get c. Go to step 8.

4)   if (cnt_1 = 12) and (cnt_2 + cnt_3 = 4), then replace the locations corresponding to (cnt_4) and (cnt_3) with 2^{m-1} preserving the sign. Replace the others with zero.

     If cnt_3 = 0, then set modify flag true. Perform IWHT to obtain c. Go to step 7.

5) If $(\text{cnt}\_1 + \text{cnt}\_2 = 16)$ then replace all the components of R with 2^{m-2} preserving the sign. Perform IWHT to obtain c.

6) Set the uncorrect flag true. Go to step 8.

7) If modify flag is true, assign $c \longrightarrow c_1$. Repeat step 5 and obtain c_2. Check for validity of c_1 and c_2. If one of them is invalid then set the valid codeword estimate to c. If both are valid then choose the one minimum distant from r. If both are invalid set uncorrect flag true.

8) The code vector is further processed to get information bits. If uncorrect flag is true declare that uncorrectable error has happened.

3.5 WHT domain decoding of $R(2,2^m)$

We are now in a position to give the new WHT domain decoding algorithm for $R(2,2^m)$ based on WHT domain decoding algorithms developed in the previous section. We shall present two decoding algorithms one of which has less number of WHT computations. We first consider the simpler of the two decoding algorithm that requires more number of computations of WHT. We start with the following lemma giving an alternate decomposition of a larger RM code into smaller RM subcodes in a different way from the decomposition of the TSKN algorithm.

Lemma 3.5.1 : In the following A_v denotes a codeword belonging to $RM(2,2^v)$ and B_v denotes a codeword belonging to $RM(1,2^v)$ and $|0|B_v|$ denotes the concatenation of all 0 vector (length 2^v) and B_v. Let $p^*(c)$ denote the concatenation $|c|....|c|$ (p times). We have

$$A_m \cong |0|B_{m-1}| \oplus 2^*(|0|B_{m-2}|) \oplus 2^{2}*(|0|B_{m-3}|) \oplus ...$$
$$\oplus 2^{v-1}*(|0|B_{m-v}|) \oplus ... \oplus 2^{m-5}*(|0|B_4|) \oplus 2^{m-4}*(A_4)$$

Proof: Invoking the the concatenation property of of RM codes, A_m splits into

$$A_m = |A_{m-1} | A_{m-1} \oplus B_{m-1}| = |0 | B_{m-1}| \oplus 2^*|A_{m-1}|$$

The proof follows from repeated application of the concatenation property.   □

Let us now consider examples illustrating the new way of decomposition.

Example 3.5.1: Consider second order code $R(2,2^5)$ of length 32. There is only one stage in the decomposition as against 3 in TSKN method (See Figure 3.2.2). In Figure 3.5.1 A ϵ $R(2,2^4)$ and B ϵ $R(1,2^4)$. Note that code B has the same error-correcting capability as $R(2,2^5)$.
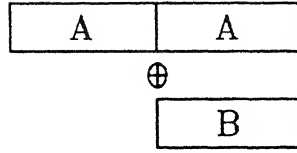
Figure 3.5.1   Decomposition of $R(2,2^5)$

Example 3.5.2: The following is second order RM code of length 64. The decomposition has 3 stages as against 6 stages in the case of TSKN algorithm (See Figure 3.2.1). 'A' denotes code belonging to $R(2,2^4)$ and 'B_2' denotes the code belonging to $R(1,2^4)$ and B_1 denotes the code belonging to $R(1,2^5)$.
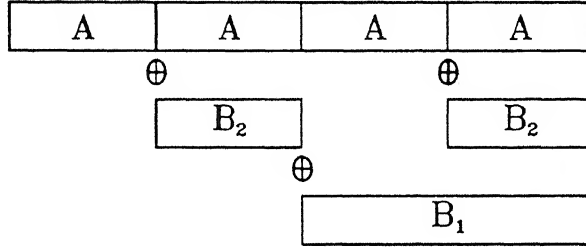
Figure 3.5.2   Decomposition subcodes of $R(2,2^6)$

Remark 3.5.1 : It is evident that the decomposition results in smaller RM codes consisting of B_i's, for i=4 to m−1, and $2^{m-4}*A_4$. For the purpose of decoding the subcode B_i's, for i = 1 to m−4, can be concatenated to make it a decoding of $R(1,2^{m-1})$. Therefore decoding of $R(2,2^m)$ is decomposed into decoding of B_j's based on WHT domain decoding of $R(1,2^{m-1})$ and decoding of $2^{m-4}*A_4$ by the WHT domain method proposed in Section 3.4. Note that the number of stages in the decomposition is less compared to decomposition given in TSKN algorithm and this may reduce the storage requirements of the WHT domain decoding algorithm for $R(2,2^m)$. Another contrast is that unlike in the TSKN algorithm, not all of the RM subcodes are of SEC−DED type.

WHT Domain Decoding Algorithm−I for $R(2,2^m)$

1) Let $r=|r_1|r_2|$ be the received vector. Obtain $r' = r_1 \oplus r_2 \in B_{m-1}$. Decode r' using WHT decoder for $R(1,2^{m-1})$ and obtain c'. Modify the received vector as $r=|r_1|r_2 \oplus c'|$.

2) Divide $r=|r_1|r_2|r_3|r_4|$. Obtain $r'' = |r_1 \oplus r_2|r_3 \oplus r_4| = |c'' \oplus e'|c'' \oplus e''|$. Decode r'' using WHT decoder for $R(1,2^{m-1})$ and obtain c''. Modify the received vector as
$$r=|r_1|r_2 \oplus c''|r_3|r_4 \oplus c''|.$$

3) In general for v^{th} step, where 'v' varies from m−1 to 4, divide r as

$$r = |r_1|r_2|\ldots|r_{2^{m-k}-1}|r_{2^{m-k}}|\ldots|r_{2^{m-v}-1}|r_{2^{m-v}}|$$

Adding consecutive sub−blocks and concatenating we obtain

$$r_{m-v} = |r_1 \oplus r_2|\ldots|r_{2^{m-k}-1} \oplus r_{2^{m-k}}|\ldots|r_{2^{m-v}-1} \oplus r_{2^{m-v}}|$$

which is the same as

$$r_{m-v} = |c_{m-v} \oplus e_1|\ldots|c_{m-v} \oplus e_2|\ldots|c_{m-v} \oplus e_{2^{m-v}-1}| \in R(1, 2^{m-1}).$$

Decode using WHT decoder for $R(1, 2^{m-1})$ and obtain c_{m-v}. Modify the received vector as

$$r = |r_1 | r_2 \oplus c_{m-v} |\ldots|r_{2^{m-k}-1} | r_{2^{m-k}} \oplus c_{m-v}|\ldots|r_{2^{m-v}-1} | r_{2^{m-v}} \oplus c_{m-v}|$$

4) At this stage we have

$$r = |r_1|r_2|\ldots|r_{2^{m-4}-1}|r_{2^{m-4}}|$$

where $r_i = c_i \oplus e_i$, where $c_i \in R(2, 2^4)$. c_i is decoded using WHT decoding algorithm for $2^{m-4} * R(2, 2^4)$ given in the Section 3.4.1.

Let us consider the same received word of Example 3.3.1 and decode it using the WHT based method.

Example 3.5.3 : Consider a RM code word of length 32 and order 2. Its dimension is 16 and minimum distance is 8. So it has an error correcting capability of 3. Choose the following code word $c = |c_1|c_2|$ for the transmission, where

$c_1 = \quad 0\,0\,1\,1\,|\,0\,0\,1\,1\,|\,0\,0\,1\,1\,|\,0\,0\,1\,1$

$c_2 = \quad 0\,1\,0\,1\,|\,0\,1\,0\,1\,|\,0\,1\,0\,1\,|\,0\,1\,0\,1$

Suppose the error pattern is as

$e_1 = \quad 0\,0\,0\,0\,|\,0\,0\,0\,1\,|\,0\,0\,0\,0\,|\,0\,0\,0\,0$

$e_2 = \quad 0\,0\,0\,0\,|\,1\,0\,0\,0\,|\,0\,0\,0\,1\,|\,0\,0\,0\,0$

In WHT domain method there are two levels of decoding as against 3 in Example 3.3.1. First step is to decode B as shown in the Fig. 3.5.3.

i) Decoding of B: The regenerated vector corresponding to B is given by $r_1 = |c_1 \oplus c_2 \oplus e_1 \oplus e_2|$. Then

$r_1 = 0\,1\,1\,0\,|\,1\,1\,1\,1\,|\,0\,1\,1\,1\,|\,0\,1\,1\,0$
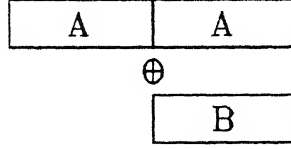
Figure 3.5.3   Decomposition of $R(2,2^5)$

The WHT of r_1 is given by

$$R_1 = -6\ 2\ 2\ 10\ |\ 2\ 2\ 2\ 2\ |\ -2\ -2\ -2\ -2\ |\ 6\ -2\ -2\ 6$$

Since the highest valued component is at position 3, the codeword B is obtained as the inverse WHT of

$$0\ 0\ 0\ 16\ |\ 0\ 0\ 0\ 0\ |\ 0\ 0\ 0\ 0\ |\ 0\ 0\ 0\ 0$$

or $B = 0\ 1\ 1\ 0\ |\ 0\ 1\ 1\ 0\ |\ 0\ 1\ 1\ 0\ |\ 0\ 1\ 1\ 0$

ii)   Decoding of A

Now subtracting the contribution of B, we have

$$r = 0\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 0\ |\ 0\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 1$$

$$0\ 0\ 1\ 1\ |\ 1\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 0\ |\ 0\ 0\ 1\ 1$$

Calculating the 16−point WHT of two halves of r separately and adding we have

$$2\ -6\ 26\ 2\ |\ 2\ 2\ 2\ -2\ |\ -2\ -2\ -2\ -2\ |\ -2\ 6\ 6\ -2$$

Since 26 > 22, threshold_3, the code spectral pattern corresponding to this is given by

$$0\ 0\ 16\ 0\ |\ 0\ 0\ 0\ 0\ |\ 0\ 0\ 0\ 0\ |\ 0\ 0\ 0\ 0$$

and therefore A is given by

$$A = 0\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 1$$

Remark 3.5.2 : Note that the Algorithm−I does not give any procedure for obtaining the information bits. One way is to re−combine the sub codewords to form the main codeword which is then processed to obtain the information bits in the usual manner. Another elegant way to obtain the information bits, exploiting the superimposition property, is described in the second version of the algorithm.

Remark 3.5.3 : The Algorithm−I for $R(2,2^m)$ requires two WHT to be done at each stage. The first pre−process WHT is required before the processing by the WHT based decoder and the second post−process WHT to get the decoded time domain code vector for further processing. The Algorithm−2 for $R(2,2^m)$, in addition to easing the process of recovering the information bits, also alleviates the need to do post−process WHT.

Remark 3.5.4: We have seen that decoding of B_i's can be concatenated to give a codeword of $R(1,2^{m-1})$ and this symmetry is useful in designing the decoder architecture. Moreover, the codeword of $R(1,2^{m-1})$

becomes highly structured with each stage. It is known that the WHT of $|c|c|$ is of the form $|C|0|$. Therefore as each stage is decoded, the number of comparisons to find the highest valued spectral component decreases by half. For example at the first stage all 2^{m-1} positions have to be searched to find the maximum valued spectral component. This reduces to searching first 2^{m-2} positions at the second stage and so on.

An improved decoding algorithm for $R(2,2^m)$

The main computational overhead the algorithm-1 for decoding of $R(2,2^m)$ is the computation of forward and inverse WHT transforms at each stage of the decoding algorithm and that of recovering the information bits. The basic idea of improvization consists in reordering of the generating vectors of $R(2,2^m)$. Such an encoding procedure not only speeds up the decoding but also simplifies the process of recovering the information bits. The price to be paid is the moderate memory for storing the basis vectors of $R(2,2^m)$. Our starting point is the set of $"{}^mC_2 + {}^mC_1 + 1"$ generating vectors $1, v_1, v_2, ..., v_m, v_{1,2}, ..., v_{1,m}, ..., v_{m-1,m}$ (Notation $v_{i,j}$ denotes $v_i . v_j$) of $R(2,2^m)$ and the objective is to obtain a suitable reordering of the generating vectors for efficient encoding and decoding.

In order to motivate the discussion, let us closely look at the structure of the sub-codewords obtained at each step of the decoding Algorithm-1. The nature of the sub-codeword obtained at the end of step 1 of the decoding algorithm-1 is of the form $|0|c|$, where each block is of length 2^{m-1}. Note that 'm' generating codewords of $R(2,2^m)$, namely, $v_m, v_{1,m}, ..., v_{m-1,m}$ are of the form $|0|c|$ and consequently, codewords of the vector space V_1 spanned by these generating code vectors have similar structure. So 'm' of the information bits can be used to specify the code subspace V_1. Similarly sub-codeword obtained at the end of step 2 of the decoding algorithm is of the form $|0|c|0|c|$ where each sub-block is of length 2^{m-2}. Consider the subspace generated by the set of 'm-1' codewords $v_{m-1}, v_{m-1,1}, ..., v_{m-1,m-2}$. It is clear that these codewords do not belong to V_1 and by themselves span the subspace V_2, whose vectors are of the form $|0|c|0|c|$. Therefore next 'm-1' information bits can be used to specify V_2. Continuing the above arguments, in general, sub-codeword obtained at the end of step 'r' is of the form $|0|c|...|0|c|$ (Repetition factor 2^r) and correspondingly, the structure of the codewords in the code subspace V_r generated by $v_{m-r+1}, v_{1,m-r+1}, ..., v_{m-r,m-r+1}$ is of the same form and therefore, next 'm-r+1' information bits can be used to specify V_r. The procedure of assigning the information bits is repeated for the first 'm-4' stages of the decoding algorithm. At the beginning of step 4, the sub-codeword generated is of the form $|c|c|...|c|$ (Repetition factor 2^{m-4}). It is also easy to see that upto the step 4 of the decoding algorithm-1, of the

total of $^mC_2 + ^mC_1 + 1$ information bits, only 11 bits needs to be encoded and these bits can be naturally assigned to choose a codeword from $R(2,2^4)$.

Now let us discuss how recovery of the information bits is made at the time of decoding. We have seen that first '$m-4$' steps of the algorithm-1 perform decoding of $R(1,2^{m-1})$. Consider typical decoding operations to be done at the step 'r'. The input to the WHT module is of the form $|r_1|r_2|...|r_p|$, where $p=2^r, r_k = c \oplus e_k$, for k=1 to p, and each sub-block is of length 2^{m-r-1}. It is known from decomposition Lemma 3.5.1 that the codeword c belongs to $R(1,2^{m-r-1})$. The code subspace corresponding to r^{th} decoding stage is spanned by vectors v_{m-r+1} $v_{1,m-r+1},...,v_{m-r,m-r+1}$. A close examination of the structure of the generating codewords reveals that the role of v_{m-r+1} is to facilitate proper positioning of the sub-blocks in the form $|0|c|0|c|...|0|c|$.

Now from Lemma 3.4.1 and 3.4.2 it follows that the highest valued spectral component, in the absolute sense, of the WHT of the received vector $|r_1|r_2|...|r_p|$ is contained in the first sub-block of the transform vector. Therefore comparison operations need to consider only the first 2^{m-r-1} elements of the transform vector. If the generating codewords are stored, then the index of the position of the highest valued spectral component can be used to recover the code vector $|0|c|0|c|...|0|c|$. Moreover, the "negative" sign of the highest valued component indicates that v_{m-r+1} has to be added to $|0|c|0|c|...|0|c|$. Therefore if we have moderate storage for generating $R(2,2^{m-2})$, the post-process WHT of the decoding algorithm-1 is not necessary. Moreover, part of the information bits are given by the index of the maximum valued spectral location. This is the computational advantage of the improved algorithm-2 over algorithm-1 at the expense of moderate memory to store the generating codewords of $R(2,2^m)$. We now present efficient decoding algorithm-2 for decoding of $R(2,2^m)$.

WHT Domain Decoding Algorithm-II for $R(2,2^m)$

1)   Let $r = |r_1|r_2|$ be the received vector. Obtain $r' = r_1 \oplus r_2 \in B_{m-1}$. Compute R' WHT of r' and find the location 'max_1' of R' with the maximum absolute value. Using max obtain c' from the generating matrix. Modify the received vector as $r = |r_1|r_2 \oplus c'|$. The first 'm' bits of information are specified by max_1.

2)   Divide $r = |r_1|r_2|r_3|r_4|$. Obtain $r'' = |r_1 \oplus r_2|r_3 \oplus r_4| \in R(1,2^{m-1})$. Decode r'' as in step 1 using WHT decoder for $R(1,2^{m-1})$ and obtain c''. Let max_2 be the location of highest spectral component of r''. Modify the received vector as

$$r = |r_1|r_2 \oplus c''|r_3|r_4 \oplus c''|$$

Next (m−1) bits of information are specified by max_2.

3) In general for v^{th} step, where 'v' varies from m−1 to 4,

$$r = |r_1|r_2| \dots |r_{2^{m-k}-1}|r_{2^{m-k}}| \dots |r_{2^{m-v}-1}|r_{2^{m-v}}|$$

obtaining 2^{v-1} copies of B_{m-v}. Concatenate to obtain

$$r_{m-v} = |r_1 \oplus r_2| \dots |r_{2^{m-k}-1} \oplus r_{2^{m-k}}| \dots |r_{2^{m-v}-1} \oplus r_{2^{m-v}}| \epsilon R(1,2^{m-1}).$$

Decode as in step 1 using WHT decoder for $R(1,2^{m-1})$ and obtain c_{m-v}. Let max_v be the location of highest spectral component of r_{m-v}. Update r as

$$r = |r_1|r_2 \oplus c_{m-v}| \dots |r_{2^{m-k}-1}|r_{2^{m-k}} \oplus c_{m-v}| \dots |r_{2^{m-v}-1}|r_{2^{m-v}} \oplus c_{m-v}|$$

Next (m−v) information bits are specified by max_v.

4) At this stage we have

$$r = |r_1|r_2| \dots |r_{2^{m-4}-1}|r_{2^{m-4}}|$$

where $r_i = c \oplus e_i$, where $c \epsilon R(2,2^4)$. c_i is decoded using WHT decoder for $2^{m-4} * R(2,2^4)$ given in the Section 3.4.1. Compute inverse Reed–Muller transform of c to get the remaining 11 information bits.

3.6 WHT based decoding algorithm for $R(3,2^m)$

We now investigate in this section extension of ideas of previous section to obtain decoding algorithm for any general $R(3,2^m)$ to decode upto the minimum distance. As explained in Section 2.3 our starting point should have been the histogram characterization of WHT spectrum of $R(3,2^m)$. However, we have already explained in Section 3.3 the difficulties of obtaining the histogram characterization of WHT spectrum of third and higher order RM codes. Moreover our experience with developing WHT based decoding algorithm for $R(2,2^m)$ is a pointer to the fact that, even if complete histogram characterization were available, they would not be of use beyond certain length. It may be recalled here that for the second order case if the length of the code is greater than 16, then there are error patterns which change a positive valued spectral component to negative ones thus rendering further processing meaningless. This fact necessitated decomposition algorithm of some sort which we could derive for second order code in the Lemma 3.5.1. A little thought shows that the decomposition of Lemma 3.5.1 holds in general for r^{th} order larger RM code in to a special r^{th} order code and $r-1^{th}$ order RM subcodes. And the special r^{th} order code is the first nontrivial r^{th} order code having minimum distance of 4 and iterated 2^{m-r} times so that the total length is 2^m. In particular, for the third order case, the first nontrivial case is that of $R(3,2^5)$ with

the minimum distance 4. This explains the choice of this specific third order code for WHT characterization.

3.6.1 WHT characterization of $R(3,2^5)$

As $R(3,2^5)$ is of smaller length it is possible to do computer analysis of histogram classification of WHT spectrum of the code and the results are presented in Table 3.6.1.

The histogram patterns of entries 1,7 and 8 of the Table 3.6.1 correspond to the subcode $R(2,2^5)$ of $R(3,2^5)$ and hence need no proof. Histogram patterns of entry 2 correspond to RM basis function of 3 variables and therefore justified by Theorem 3.3.1. The remaining 4 histogram classes are obtained by computer search. As analytical proof is likely to be tedious and do not promise any new information, we do not give any analytical proofs.

Table 3.6.1 Histogram of the RM code (32, 21, 4)

Pattern Number	Spectral Values	Histogram
1	0,32	31,1
2	0,8,24	24,7,1
3	0,8,16	22,8,2
4	4,12,20	30,1,1
5	0,8,16	19,12,1
6	4,12	28,4
7	0,8	16,16
8	0,16	28,4

Before proceeding to give WHT based decoding algorithm for $R(3,2^5)$ for correction of single error, we now verify that the conditions for recoverability of histogram pattern are satisfied. Note from the Table 3.6.1 that the minimum nonzero spectral value is 4 and that different spectral values within a histogram class differ at least by 8.

The minimum spectral value 4 can change to at most 2 if a single error occurs and this ensures that the signs of spectral value do not change in the event of an error. Therefore first condition for recoverability is satisfied. As any two spectral values of a histogram class differ at least by 8 and a single error can change a spectral value at most by 2, it follows that the range of variation of spectral values are distinct. Therefore second condition for recoverability is also satisfied.

3.6.2 WHT decoding algorithm for $R(3,2^5)$

The Table 3.6.2 shows the histogram classification for modified WHT spectrum of $R(3,2^5)$ due to a single error. The modified code spectrum has 16 different histogram classes and decoding algorithm is more

Table 3.6.2 : Modified histogram pattern of (32, 21, 4) RM code due to error
[.] in first column refers to codeword histogram class index w.r.t Table 3.6.1.

Pattern Number	Spectral Values	Histogram
1 [1]	2,30	31,1
2 [2]	2,6,10,22	24,4,3,1
3 [3]	2,6,10,14,18	22,6,2,1,1
4 [4]	2,6,14,18	18,12,1,1
5 [4]	2,6,10,22	20,10,1,1
6 [3]	2,6,10,14	22,4,4,2
7 [4]	2,6,10,18	15,15,1,1
8 [5]	2,6,10,14	19,7,5,1
9 [6]	2,6,10,14	15,13,3,1
10 [5]	2,6,10,18	19,9,3,1
11 [6]	2,6,10,14	18,10,2,2
12 [6]	2,6,10	12,16,4
13 [7]	2,6,10	16,10,6
14 [8]	2,14,18	28,3,1
15 [6]	2,6,10,14	21,7,1,3
16 [2]	2,6,26	24,7,1

complex compared to WHT decoding of $R(2,2^4)$. The key clue to identifying the histogram class is counting the number of components with the value 2.

WHT Domain Decoding Algorithm for $R(3,2^5)$

1) Compute R the WHT of real vector corresponding to $r = c \oplus e$ and obtain R_a containing absolute values of components R.

2) Find cnt_0, the number of components of R_a, having value 2; and cnt_4, the number of components of R_a, having spectral value > 2 but ≤ 6.

3) Set complete flag false.

4) If cnt_0 = 16, 19, 22, 24, 28 or 31, then substitutions to R_a are done as follows. The sign of a component of R_a is taken to be the same as that of corresponding component of R.

16: replace 2 by zero and 6 or 10 by 8.

19: replace 2 by zero; 6 or 10 by 8 and14 or 18 by 16.

22: replace 2 by zero; 6 or 10 by 8 and 14 or 18 by 16.

24: replace 2 by zero; 6 or 10 by 8 and 22 or 26 by 24.

28: replace 2 by zero; and 14 or 18 by 16.

31: replace 2 by zero; 30 by 32.

After appropriate substitutions are over, set complete flag true. Go to step 6.

5) If (complete flag = false) and (cnt_0+cnt_4) = 28 or 30, then do the following replacements to R_a and determine the sign as explained in step 4.

case 28: replace 2 or 6 by 4; and 10 or 14 by 12.

case 30: replace 2 or 6 by 4; 10 or 14 by 12 and 18 or 22 by 20.

6) Perform IWHT on R_a and obtain the transmitted code vector. The code vector is further processed to obtain the information bits.

We now proceed to develop the decoding algorithm for $2^{m-5}*R(3,2^5)$. The key clue to identifying the histogram class is again cnt_0, the number of spectral values within the range of variation for spectral value '0'. Then a possible choice for the histogram class of the received signal is taken to be that codeword histogram pattern having the same number of zeros as cnt_0. Since there are two histogram classes having least nonzero spectral value 4, there may be two possible ambiguities in the transmitted codeword spectrum. It is shown later that the number of ambiguities can be effectively reduced to 1, the same number as in the case of WHT domain decoding of $2^{m-4}*R(2,2^4)$. The minimum and maximum range of values taken by distinct spectral values of the code spectrum are shown below.

Table 3.6.3 : Range of variation for spectral values of $2^{m-3}*(32, 21, 4)$
(.) denotes spectral values of (32, 21, 4) RM code

Minimum Possible Value	Actual Spectral Value	Maximum Possible Value
$-(2^{m-3}-2)$	0	$(2^{m-3}-2)$
2	2^{m-3} (4)	$(2.2^{m-3}-2)$
$(2^{m-3}+2)$	2.2^{m-3} (8)	$(3.2^{m-3}-2)$
$(2.2^{m-3}+2)$	3.2^{m-3} (12)	$(4.2^{m-3}-2)$
$(3.2^{m-3}+2)$	4.2^{m-3} (16)	$(5.2^{m-3}-2)$
$(4.2^{m-3}+2)$	5.2^{m-3} (20)	$(6.2^{m-3}-2)$
$(5.2^{m-3}+2)$	6.2^{m-3} (24)	$(7.2^{m-3}-2)$
$(7.2^{m-3}+2)$	8.2^{m-3} (32)	8.2^{m-3}

Some observation that are useful in deriving the algorithm are obtained from a closer inspection of the above table. They are as follows:

1) It is clear from Table 3.6.3 that there is no overlap of the range of variation for two spectral values differing by 2^{m-2}. Moreover, referring to Table 3.6.1, we see that different spectral values within a histogram class differ by 8. By virtue of Lemma 3.4.1, the same spectral values differ by 2^{m-2} in the WHT spectrum of the code $2^{m-5}*R(3,2^5)$. Combining these two observations it follows that two different histogram classes, having the same least spectral value, are unambiguously distinguished. For example, it is not possible to mistake the histogram class [0 (24), 8 (7), 24 (1)] to another histogram class [0 (22) ,8 (8) ,16 (2)] as the range of variation for spectral values 0, 8, 16 and 24 are distinguishable from each other, even if codeword is corrupted by error during the transmission. Another implication is that two codewords belonging to the same histogram class are also unambiguously distinguished.

2) The least spectral values in Table 3.6.1 are either 0 or 2^{m-3}. Since there are 2 histogram classes with the least value 2^{m-3}, in theory at least, a histogram class with least spectral value 0 could be mistaken for 2 possibilities. And the previous argument shows both possibilities are not possible simultaneously. So effectively ambiguities are reduced to two cases which can be handled in the same way as explained in the decoding algorithm for $2^{m-4}*R(2,2^4)$.

We first present decoding algorithm for $2^{m-5}*R(3,2^5)$ followed by decoding algorithm for $R(3,2^m)$. For simplicity of discussion we omit details of obtaining information bits from the code vectors. We note that the method of obtaining partial information bits for efficient decoding of $R(3,2^m)$ can be obtained as straightforward extension of the method explained in algorithm–2 for $R(2,2^m)$ and therefore omitted.

<div align="center">

Decoding Algorithm for $2^{m-5}*R(3,2^5)$

(thr_1=2^{m-3}–2; thr_2=2.2^{m-3}–2)

</div>

Note : replace {A...B} by C means that substitute those values of spectrum from A to B by C.

1) Compute first 32 components of the WHT R of the real vector corresponding to r = c ⊕ e and obtain R_a containing absolute values components of R.

2) Compare components of R_a with 2 thresholds and find the following:

cnt_1: Number of components ≤ thr_1.

cnt_2: Number of components > thr_1 and ≤ thr_2.

3) Set complete flag false.

4) If cnt_1 = 16, 19, 22, 24, 28 or 31, then do the replacements to R_a as explained below. The sign of R_a is taken to be the same as that of the corresponding component of R.

$cnt\_1 = 16$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0 and $\{(2^{m-3}+2)...(3.2^{m-3}-2)\}$ by 8;
If $(cnt\_1+cnt\_2 = 28)$ then set modify1_flag true.

$cnt\_1 = 19$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0; $\{(2^{m-3}+2)...(3.2^{m-3}-2)\}$ by 8; and $\{(4.2^{m-3}+2)...(5.2^{m-3}-2)\}$ by 16. If $(cnt\_1+cnt\_2 = 28)$ then set modify1_flag true; else, if $(cnt\_1+cnt\_2 = 30)$ then set modify2_flag true;

$cnt\_1 = 22$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0; $\{(2^{m-3}+2)...(3.2^{m-3}-2)\}$ by 8; and $\{(4.2^{m-3}+2)...(5.2^{m-3}-2)\}$ by 16. If $(cnt\_1+cnt\_2 = 28)$ then set modify1_flag true; else, if $(cnt\_1+cnt\_2 = 30)$ then set modify2_flag true;

$cnt\_1 = 24$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0; $\{(2^{m-3}+2)...(3.2^{m-3}-2)\}$ by 8; and $\{(5.2^{m-3}+2)...(7.2^{m-3}-2)\}$ by 24. If $(cnt\_1+cnt\_2 = 30)$ then set modify2_flag true;

$cnt\_1 = 28$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0; $\{(4.2^{m-3}+2)...(5.2^{m-3}-2)\}$ by 16; If $(cnt\_1+cnt\_2 = 28)$ then set modify1_flag true;

$cnt\_1 = 31$: replace $\{-(2^{m-3}-2)...(2^{m-3}-2)\}$ by 0 and $\{(7.2^{m-3}+2)...2^m\}$ by 32.

For each of the above cases after respective substitution, compute IWHT of R_a to obtain c.

Set complete flag true and go to step 6.

5)   If (complete flag is false) and $(cnt\_1+cnt\_2) = 28$ or 30, then do following substitutions to R_a and determine the sign as in step 4.

case 28 : replace $\{2...(2.2^{m-3}-2)\}$ by 4; and $\{(2.2^{m-3}+2)...(4.2^{m-3}-2)\}$ by 12.

case 30 : replace $\{2...(2.2^{m-3}-2)\}$ by 4; $\{(2.2^{m-3}+2)...(4.2^{m-3}-2)\}$ by 12; and $\{(4.2^{m-3}+2)...(6.2^{m-3}-2)\}$ by 20. Go to step 7.

6)   If either modify1_flag is true or modify2_flag is true, assign c $\longrightarrow$ c_1 and obtain another codeword estimate c_2 from step 5. If one of them is invalid choose c as the valid estimate. If both are valid then decide on the basis of ML criterion. If both are invalid, then detect uncorrectable condition.

7)   The code vector is further processed to get the information bits.

WHT Domain Decoding Algorithm for $R(3,2^m)$

1)   Let $r=|r_1|r_2|$ be the received vector. Obtain $r' = r_1 \oplus r_2$, whose codeword belongs to $R(2,2^m)$. Decode r' using WHT decoder for $R(2,2^{m-1})$ and obtain c'. Modify the received vector as $r = |r_1|r_2 \oplus c'|$.

2)   Divide $r = |r_1|r_2|r_3|r_4|$. Obtain $r'' = |r_1 \oplus r_2|r_3 \oplus r_4|$ whose codeword belongs to $R(2,2^{m-1})$. Decode r'' using WHT decoder for $R(2,2^{m-1})$ and obtain c''

Modify the received vector as

$$r = |r_1|r_2 \oplus c''|r_3|r_4 \oplus c''|$$

3) In general for v^{th} step, where 'v' varies from m−1 to 5

$$r = |r_1 \mid r_2|...|r_{2^{m-k}-1} \mid r_{2^{m-k}}|....|r_{2^{m-v}-1} \mid r_{2^{m-v}}|$$

obtaining 2^{m-v} sub−blocks of length 2^v. Concatenate to obtain

$$r_{m-v} = |r_1 \oplus r_2|...|r_{2^{m-k}-1} \oplus r_{2^{m-k}}| ... |r_{2^{m-v}-1} \oplus r_{2^{m-v}}|$$

whose codeword is in $R(2,2^{m-1})$. Decode using WHT decoder for $R(2,2^{m-1})$ and obtain c_{m-v}. Modify the received vector as

$$r = |r_1|r_2 \oplus c_{m-v}|...|r_{2^{m-k}-1}|r_{2^{m-k}} \oplus c_{m-v}|...|r_{2^{m-v}-1}|r_{2^{m-v}} \oplus c_{m-v}|$$

4) At this stage we have

$$r = |r_1|r_2|...|r_{2^{m-5}-1}|r_{2^{m-5}}|$$

where $r_i = c \oplus e_i$, where $c \epsilon R(3,2^5)$. r is decoded using WHT decoder for $2^{m-5} * R(2,2^5)$ described above.

Example 3.6.1 : Consider the following corrupted codeword of length 64. The particular code is $R(3,2^6)$ which can correct upto three errors.

                1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0
                0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1
                0 0 1 1 1 0 0 1 0 0 1 0 1 1 0 0
                0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1

As per the decoding algorithm it involves single decoding of $R(2,2^5)$ which can be done as explained in Section 3.5. The second order code is decoded to be

                1 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0
                0 0 1 1 1 1 1 1 0 1 0 0 1 1 0

Therefore input to third order decoder is

                1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0
                0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1
                1 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1
                0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1

The WHT is found to be

18 10 −14 −14 2 2 2 −6 14 −18 14 −18 −2 6 −2 −2

2 2 2 −6 −14 10 18 −14 −2 6 −2 −2 −18 −18 −18 −10

The codeword spectrum is found to be

16 16 −16 −16 0 0 0 0 16 −16 16 −16 0 0 0 0

0 0 0 0 −16 16 16 −16 0 0 0 0 −16 −16 −16 −16

The corresponding time domain vector is

1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0

0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1

3.7 WHT based decoder architectures for R(2,2^m) and R(3,2^m)

We first propose decoder architectures for implementing WHT domain decoding algorithms for R(2,2^m) and R(3,2^m). Then we give expression for decoding delay in terms of NOR gate delay as was done in [46]. It is shown for m ≥ 8 our method has significantly lesser decoding delay compared to TSKN method.

Decoding of R(2,2^m) involves two main steps: (m−4) − fold WHT domain decoding of R(1,2^{m-1}) and a single stage WHT domain decoding of 2^{m-4}*R(2,2^4). The decoder operations in the case of R(1,2^{m-1}) consists of decomposition, computation of the WHT transform, finding the highest valued spectral component, substitution, computing the inverse WHT and finally regeneration. For the purpose of decoding it is convenient to represent the binary bits by ±1 in sign−magnitude form and adders required for computation of WHT are appropriately chosen. Whenever decoder processing requires the binary 0 or 1 version, it is enough to take the sign bits. The butterfly algorithm is used for fast computation of the WHT employing 2^m (n) adders 'm' (log n) times. The maximum absolute valued component is found employing n comparators in log n time. The regeneration and decomposition steps are done as in TSKN method in a single step using n/2 EX−OR gates. Substitution can be done in constant time involving a few logic gate delays.

Decoding architecture for the first stage of decoding algorithm for R(2,2^m) is shown in Figure 3.7.1. In the first stage, first order RM subcodes are decoded. This stage proceeds in (m−4) steps, j^{th} step consisting of 2^{j-1} modules 'Λ_j', for j=1 to (m−4), performing the regeneration and decomposition. Module Λ_j processes vector of length 2^{m-1-j}. At j^{th} step regeneration and decomposition are done concurrently. The serial processing at j^{th} step includes WHT computation, maximal spectrum locator unit, substitution and Inverse WHT computation. Substitution is simple to implement as only one spectral value is nonzero. Note that it is possible to have a single WHT and maximum locator unit, although they have been

duplicated for simplicity. Another interesting thing to note is that the index of the maximum valued component gives the information bits directly.

Figure 3.7.2 shows the decoder architecture for performing second stage of decoding algorithm for $R(2,2^m)$, consisting of decoding of $2^{m-4}*R(2,2^4)$. The input is the first 16 coefficients of WHT of first stage decoded received vector. Each of these 16 spectral values are compared with three thresholds corresponding respectively to maximum variation of spectral values 0, 2^{m-3} and 2^{m-2}. The 4 outputs of the comparator unit for a received spectrum 's' are given by $s \leq thr\_1$, $thr\_1 < s \leq thr\_2$, $thr\_2 < s \leq thr\_3$ and $s > thr\_3$. The respective outputs from all the 16 comparators are grouped together to form $\{T_k\}$ for k=1 to 4. The values of cnt_1, cnt_2 and cnt_3 and cnt_4 are determined using parallel counters from $\{T\}$. The counting of thresholds is accomplished using parallel counters employing 3–state, 7 bit–slice Wallace tree adders [86]. The $\{T\}$ are also fed to substitution network that takes care of proper substitution range for all 16 coefficients of received spectrum. The cnt_1 and cnt_2 are used by the simple combinational logical network to determine the histogram class. The cnt_3 output, if nonzero, disables the repeat circuitry logic. The outputs of the combinational logic circuitry perform final substitution. This is followed by the circuitry for the inverse WHT computation. Note that a test is needed to check if the transformed vector represents the valid binary vector. This is easily accomplished by checking if each of the components has absolute value 1. The output of IWHT sets repeat flag which activates repeat operation if (cnt_1 = 12) and (cnt_3 > 0). Also if there is repeat operation, then provision has to made for storing the first code word estimate. If the second codeword estimate is also valid, then the two code vector estimates are compared with the received vector and the one least distant from the received vector is chosen. The information retriever performs inverse binary Reed–Muller transform to get information symbols.

Figure 3.7.3 shows only some of the details of the decoder architecture for $2^{m-5}*R(3,2^5)$, the final stage sub–decoding in WHT domain $R(3,2^m)$ decoding. The remaining decoder circuitry for $R(2,2^5)$ decoding is similar to the one described above and therefore omitted. First thing to be noted is that separate comparator unit is required for each of first 32 values of WHT of first stage decoded received vector. A comparator unit compares the received spectrum with 7 thresholds set to the maximal variation of 7 distinct spectral values $k2^{m-3}$ for k=0 to 6, in the WHT spectrum of $2^{m-5}*R(3,2^5)$. For a spectral value s, the 8 outputs generated correspond to: $s \leq thr\_1$, $thr\_k < s \leq thr\_(k+1)$ for k = 1 to 6, and $s > thr\_7$. As before the corresponding outputs of 32 comparators are grouped together of which only max_0 and max_4 are counted and used in the further processing. The remaining 6 are required for performing proper substitution.
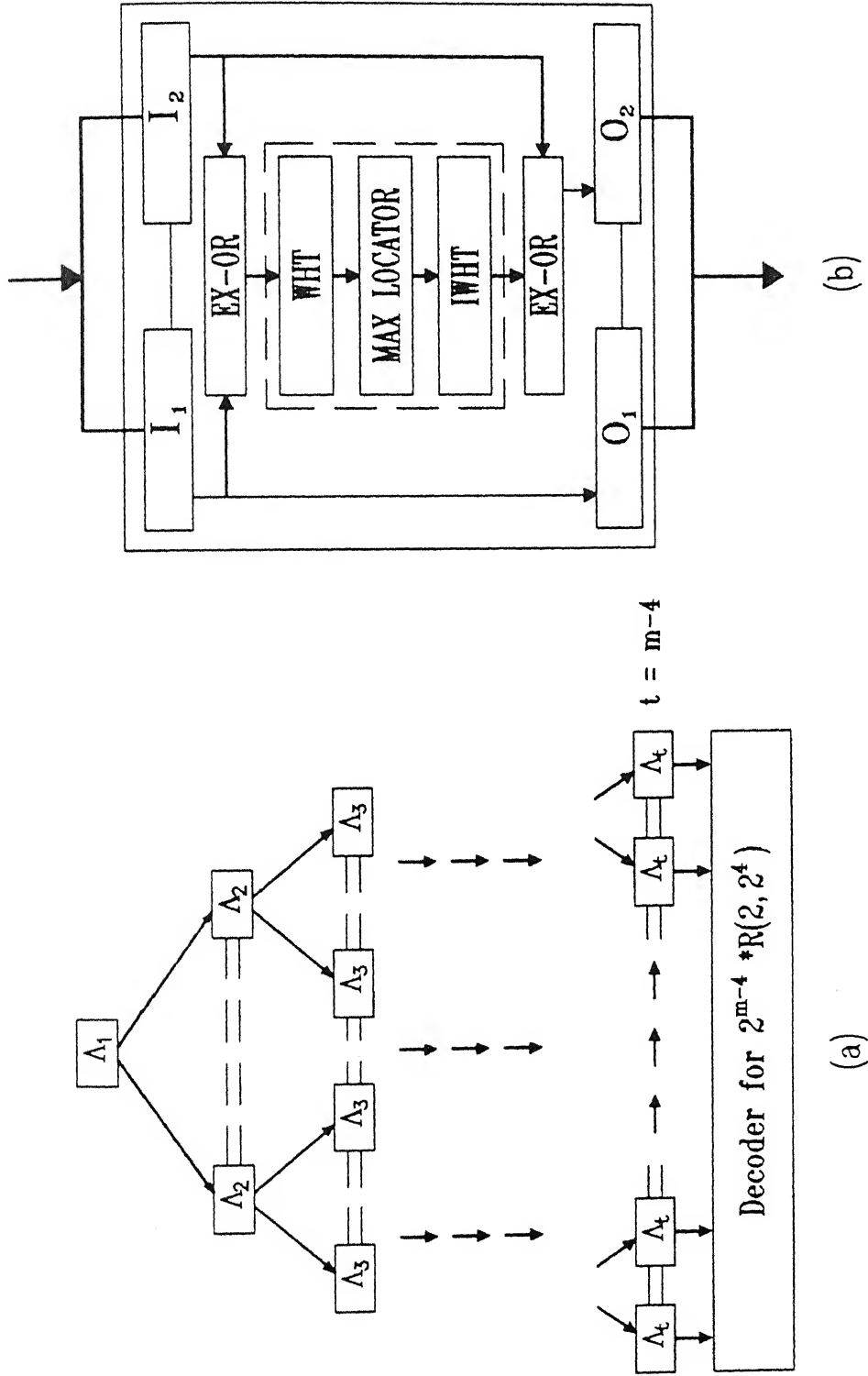
Figure 3.7.1: Decoder Architecture for First Stage WHT Decoding of $R(2, 2^m)$ (a) Overall Configuration (b) Module Λ_j (Lengths of I and 0 are 2^{m-1-j})
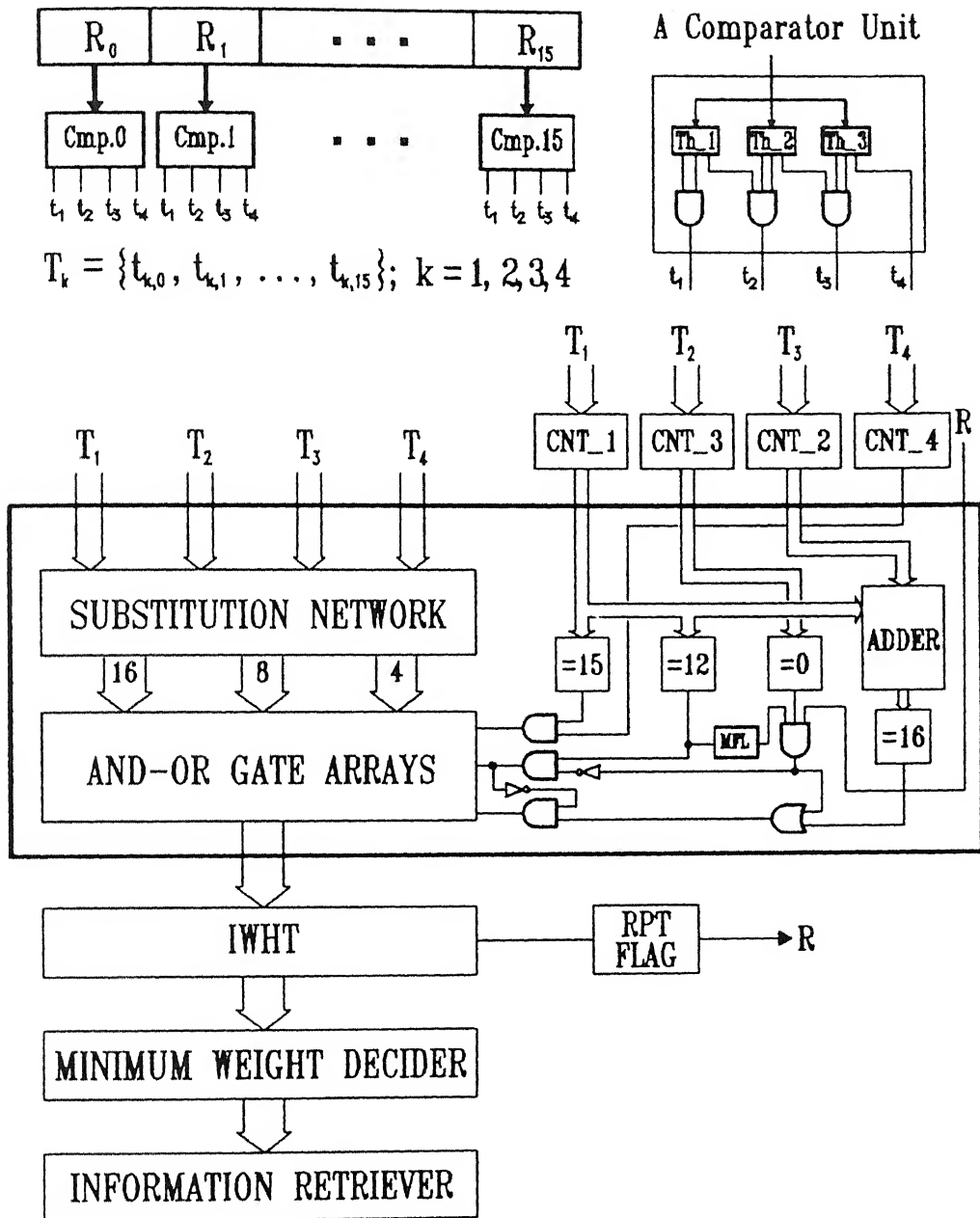
Figure 3.7.2: WHT Domain Decoder
Architecture for $2^{m-4} * R(2, 2^4)$

3.7.1 Delay analysis of TSKN algorithm

The decoding delay is measured by the delay time Δ of a NAND gate. Table 3.7.1 shows typical times gate functions and delay times.

Table 3.7.1 Typical Gate Functions and Delay times

Gate Function	Delay time
NAND	Δ
NOR	Δ
NOT	Δ
AND	2Δ
OR	2Δ
EX–OR	3Δ
EX–NOR	3Δ

From Theorem 3.2.2, it follows that there are r steps of subcode decoding, i.e. log n steps, each step involving decoding of $2^{m-r-2}*R(i-2, 2^j)$, for i = 2 to r+2, n(i) times. Then we have

$$n(i) = \binom{m-1-i}{r+2-i} \text{ for } i=2,3..r+2$$

and the decoding delay $T(r,2^m)$ is given by

$$T(r,m) = \sum_{i=0}^{r} n(i+2) \left[\tau(i+2, m-r-2) + 6\Delta \right]$$

where $\tau(s,u)$ is the decoding delay for $2^u*R(s-2,2^s)$

The term 6Δ accounts for two EX–OR gate delays accounting for decomposition and regeneration. As noted in Section 3.2 $\tau(s,u)$ delay is $2^{m-r-2}*t_d$ delay.

The computation of "t_d" consists of delay time for computing parity check bits, the delay to determine the error position or perform double error detection, the time required to compute the Hamming distance of the estimated code vector and the received vector and the time required to compare the Hamming weight with the threshold di/2, where d is the minimum distance of the code and i is the repetition factor. Computation of the Hamming weight is facilitated by a parallel counter employing 3–state output and 7–bit–slice Wallace tree as the basic unit. Then the expression for t_d becomes

$$\tau(s,u) = \left[2^u \left\{ 3s + 3\lceil \log_2(s+1) \rceil + 3\lceil \log_7 2^{u+s} \rceil + 3\lceil \log_2(2^{u+s}+1) \rceil - 5 \right\} + 3\lceil \log_5(u+s+2) \rceil \right] \Delta$$

Remark 3.7.1 : One point we would like to make in connection with the above expression for $\tau(s,u)$ is that the time for comparison with the threshold (d*i/2) is counted only once. Actually while decoding several comparisons may have to be made and this is not taken into account in the expressions for $\tau(s,u)$. We have

not given corrected expression as full details of the algorithm is not given in [46]. This additional factor slightly increases delay slightly, but otherwise not significant. The following tables show the delay for second and third order RM codes as given in [46]. Note that in the table $T'(r,2^m)$ refers to modified TSKN algorithm which has less delay and more complexity compared to TSKN algorithm [46]. For comparative purposes, we also quite corresponding delay for the case of majority logic decoding using Chen's method [79].

Table 3.7.2a : Decoding delay for $R(2,2^m)$ using TSKN Algorithm

m	$T(r,2^m)$	$T'(r,2^m)$	$T_{maj}(r,2^m)$
5	225	225	1860
6	864	740	4914
7	2960	2180	11811
8	9684	6060	27540
9	28668	15748	64386
10	82000	40540	144243
11	233520	101556	319332
12	616476	243404	933660

Table 3.7.2b : Decoding Delay for $R(3,2^m)$ using TSKN Algorithm

m	$T(r,2^m)$	$T'(r,2^m)$	$T_{maj}(r,2^m)$
6	335	335	5103
7	1552	1428	12954
8	6256	1428	12954
9	23444	5216	32130
10	78880	53040	174933
11	252720	155980	393024
12	792096	440192	1130220

3.7.2 Delay analysis of WHT domain decoding algorithm

We now derive an estimate of the delay for both second and third order RM codes with regard to the decoder architecture schematic discussed above. The estimates are correct upto a constant delay factor depending upon device characteristic. The signed magnitude representation is employed and ripple carry adders are used. The corresponding binary representation is taken from sign bits. The signed magnitude also facilitates the decomposition and regeneration by means of AND gates. It may be possible to improve delay using more sophisticated components. We have

$$T(2,2^m) \quad =(m-4)T(1,2^{m-1}) + T[2^{m-4}*R(2,2^4)]$$
$$T(3,2^m) \quad =(m-5)T(2,2^{m-1}) + T[2^{m-5}*R(3,2^5)]$$
$$=(m-5)\{(m-5)T(1,2^{m-2}) + T[2^{m-5}*R(2,2^4)]\} + T[2^{m-5}*R(3,2^5)]$$

where $T(r,2^m)$ is the WHT domain decoding time for r^{th} order RM code.

Expression for $T(1,2^m)$

$$T(1,2^m) = t_{decom} + t_{wht} + t_{comp} + t_{subs} + t_{iwht} + t_{reg}$$

$$= [2 + m(2m+13) + 3m\lceil \log_5(m+2)\rceil + 3\lceil \log_2(m+1)\rceil + 4 + m(2m+13) + 2]\Delta$$

$$= (4m^2 + 26m + 3m\lceil \log_5(m+2)\rceil + 3\lceil \log_2(m+1)\rceil + 8)\Delta$$

Expression for $T[2^{m-4}*R(2,2^4)]$

$$T[2^{m-4}*R(2,2^4)] = t_I + t_{II} + t_{III}$$

where, t_I is the time taken for the second order decoder starting from WHT computation upto the computation of max_0 and max_4 which are required to select the appropriate histogram class for substitutions to be made. t_{II} is the total time taken by the substitution unit to perform appropriate substitutions depending on the selection of combinational logic unit, time to performing inverse WHT, the time to determine the distance of estimated vector from the received vector, and time to repeat the above 3 steps if the modify flag is set. Finally t_{III} is the time taken for comparison and selection of estimated code vector least distant from the received vector and also re-encoding time for the computation of inverse Reed-Muller transform. We have

$$t_I = t_{wht} + t_{thr-comp} + t_{sum}$$

$$t_{II} = t_{comb.log} + 2(t_{aog} + t_{iwht} + t_{ex-or} + t_{ham.wt}) + t_{rpt-comb-log}$$

$$t_{III} = t_{thr\_comp} + t_{aog} + t_{re-encd}$$

The delay of a $(2^m,m)$ parallel counter is given by $3*\lceil \log_7 2^m\rceil + 3*\lceil \log_2(2^m+1)\rceil$. The purpose of a parallel counter is to compute the Hamming distance of a binary vector. From the inspection of the decoder circuit and using Table 3.7.1 the constant factors is estimated to be 100 Δ. Finally we have

$$T[2^{m-4}*R(2,2^4)] = 3m(2m+13) + 6\lceil \log_5(m+2)\rceil + 6\lceil \log_5 2^m\rceil + 6\lceil \log_7(2^m+1)\rceil + 100]\Delta$$

Expression for $T[2^{m-5}*R(3,2^5)]$

$$T[2^{m-5}*R(2,2^5)] = t_I + t_{II} + t_{III}$$

where t_i's are as defined for second order decoder case. The only difference is that threshold comparators and correspondingly substitution units are more complex. The expression becomes

$$T[2^{m-5}*R(2,2^5)] = [3m(2m+13) + 6\lceil \log_5(m+2)\rceil + 6\lceil \log_5 2^m\rceil + 6\lceil \log_7(2^m+1)\rceil + 200]\Delta$$

Table 3.7.3: WHT domain decoding delay for $R(2,2^m)$

m_value	Delay
4	400
5	708
6	1176
7	1804
8	2628
9	3693
10	4984
11	6543
12	8400

Table 3.7.4 : WHT domain decoding delay for $R(3,2^m)$

m_value	Delay
5	599
6	1430
7	3197
8	6392
9	11645
10	19757
11	31367
12	47453

Remark 3.7.1 : The difference in delay factors in TSKN and WHT domain decoding algorithms is due to different decomposition algorithms and different approaches to decoding repeated subcodes. TSKN algorithm attempts to decode each repeated block and then ascertains the validity of decoding. Therefore, the number of attempts may vary from one to as many as the number of iterated blocks. Note that the number of iterations is 2^{m-r-2}, exponential in m. WHT domain decoding algorithm, on the other hand, tries to decode in a single step. If there are any ambiguities then it requires one more decoding to complete the decoding. Therefore, as m increases WHT domain decoding delay is less compared to delay in TSKN algorithm.

Chapter 4

SPECTRAL DOMAIN DECODING ALGORITHMS
FOR 2-D BCH CODES

The zeros of a linear, t–error correcting, BCH code can be visualized as a continuous strip of 2t positions along a line. This visualization can be generalized to a linear code having a contiguous array of zeros in two dimensions, and codes so obtained are called 2–D BCH codes. These codes are used for transmission over communication channels where errors occur in the form of multiple–low–density bursts. As the zero locations of these codes have simple geometrical structure, decoding algorithm can be easily visualized for them. Such a class of codes accordingly provides a good starting point for illustrating the basic ideas of 1–D deconvolution approach to DFT domain decoding.

The notion of 2–D BCH codes is implicitly contained in the class of dual product codes studied by Chien and Ng [80]. They have outlined a general procedure for decoding dual product codes based on the fact that their parity–check code (or dual code) is also a product code. In particular, they have given a decoder architecture for the class of product codes whose component codes are majority logic decodable. As regards the error correction capability of such codes, they have found that although such codes are good for correcting multiple low–density burst errors, they are poor in correcting random errors.

Alternatively, 2–D BCH codes can be thought of as a class of 2–D cyclic codes, for which the finite field transform methods for studying the structure of the codes are available. Blahut took this approach to study 2–D BCH codes and showed that the minimum distance of 2–D BCH codes having a contiguous array of 2t x 2t zeros is at least 2t+1. He also gave a decoding algorithm for correcting upto 't' random errors using the spectral domain approach [2]–[4] (We refer to Blahut's algorithm for random error correction as BA–1). The decoding algorithm proposed here requires, as an important sub–algorithm, the Berlekamp–Massey algorithm (B–M algorithm) for synthesizing a LFSR for a given syndrome sequence along a row or column. This is followed by recursive extension of row or column syndromes along that row or column. Therefore, the implementation complexity of the decoder is little more compared to that of BCH decoder. Another feature of the decoding algorithm is that the syndrome computations for either all the rows or all the columns can be done in parallel. Blahut has also given a decoding algorithm for correction of some burst–errors with a slight modification of BA–1 (which we refer to as BA–2).

In this chapter, we give an alternative exposition of BA–1 for random errors that gives more insights into the working of the decoding algorithm. It is shown that the number of passes through B–M algorithm to find the common row or column connection polynomial is upper bounded by 't' which makes decoding more efficient compared to BA–1 and BA–2. Other minor modifications to both BA–1 and

BA–2 are also suggested which save some computations. We then present improved Blahut algorithms (called IBA–1 and IBA–2) incorporating these changes.

We employ 2–D complexity theorem to determine burst–error correcting capability of 2–D BCH codes as afforded by the defining zero locations. It is then shown that Blahut's decoding algorithm is best possible for the class of 2–D BCH codes in that it is possible to correct upto the error–correcting capability of the code. A new class of Γ 2–D BCH codes having higher rate than the class of 2–D BCH codes is introduced and they are shown to have the same error–correcting capability as that of class of 2–D BCH codes. It is not possible to use BA–1 and BA–2 directly for decoding of Γ 2–D BCH codes, but IBA–1 and IBA–2 can be used. Moreover, IBAs are shown to be the best possible algorithms for decoding of Γ 2–D BCH codes.

We also consider an alternative approach for decoding of 2–D BCH codes employing Feng–Tzeng (FT) algorithm for multiple LFSR synthesis. Finally, we consider a class of t–random error–correcting (t–REC) 2–D BCH codes and give a decoding algorithm for correcting upto designed distance. This study also helps to evaluate intuitively the constraints on the zero locations for correcting various types of random and burst errors.

Section 4.1 reviews different types of 2–D bursts so far reported in the literature. It is found convenient to define two special types of burst errors, namely quasi–cyclic 2–D bursts with area t x t and Blahut's 2–D burst with area t x t, which can be corrected using spectral decoding methods for 2–D BCH codes. Section 4.2 reviews the formulation of 2–D BCH decoding problem as given by Blahut and presents an alternative exposition based on 1–D deconvolution approach to DFT domain decoding. Some improvements to BA–1 are suggested. In Section 4.3, it is shown that BA–1 is also inherently capable of correcting burst errors, namely, quasi–cyclic burst–errors upto area t x t, in addition to simultaneously correcting upto t random errors. Some minor improvements to BA–2 are also suggested. Section 4.4 gives improved Blahut's decoding algorithms (IBA–1 and IBA–2) for correction of various random and burst errors. Section 4.5 applies 2–D complexity theorem to determine the error–correcting capabilities of the class of 2–D BCH codes, as afforded by the defining zero arrays of the code, with regard to both quasi–cyclic bursts and Blahut's bursts. It is shown that both BA–1 and BA–2 are best possible algorithms for correction of various combinations of random errors and burst–errors in that they correct upto the error–correcting limit of the code. In Section 4.6 a new class of Γ 2–D BCH codes having higher rate than the class of 2–D BCH codes is introduced. Error–correcting capabilities of the class of Γ 2–D BCH codes are determined and it is shown that IBA is best possible algorithm to decode Γ 2–D BCH codes upto their error correcting capability. Section 4.7 presents mixed time and spectral domain implementations of BA–1, BA–2 and IBA–1 and IBA–2. Section 4.8 discusses an alternative way of decoding 2–D BCH codes using multiple LFSR synthesis algorithm proposed by Feng and Tzeng [FT

algorithm]. Section 4.9 considers a class of t–random error–correcting (t–REC) 2–D BCH codes along with their decoding algorithm.

4.1 Nature of 2–D burst–errors

We have seen that both the code word and the error are taken as a n_1 x n_2 matrix over GF(q) and that the effect of error is modeled as addition over GF(q). Imai has defined the burst–error size as the area of minimum submatrix b in e, which contains non–zero components of an error e [81]. The starting position of the burst–pattern is referred to by the position of the left upper corner of b in e. In the terminology of two variate polynomials, b is represented by a polynomial b(x,y) satisfying

$$b(x, 0) \neq 0; \quad b(0, y) \neq 0;$$
$$\deg_x b(x,y) = b_1 - 1; \quad \deg_y b(x,y) = b_2 - 1.$$

Taking the position of this pattern as (i,j), the error e is expressed as

$$e(x,y) = (x^i y^j b(x,y)) \text{ modulo } (x^{n_1} - 1, y^{n_2} - 1).$$

If $0 \leq i \leq n_1 - b_1$ and $0 \leq j \leq n_2 - b_2$ then e is called b_1 x b_2 2–D burst–error. If $0 \leq i \leq n_1 - 1$ and $0 \leq j \leq n_2 - 1$, then it is called b_1 x b_2 2–D cyclic burst. Note that it is possible for a 2–D cyclic burst to wrap around in two dimensions due to the nature of modulo polynomial operations. It is shown in [82] that all bursts of area b_1 x b_2 have unique patterns and unique starting positions if $n_1 \geq 2b_1 - 1$ and $n_2 \geq 2b_2 - 1$. An expression for distinct number of b_1 x b_2 cyclic 2–D bursts in a n_1 x n_2 array is also given in [82]. A partial order relation on the set of 2–D error–bursts is defined as follows. A 2–D burst with area b_1' x b_2' is said to be less than or equal to a 2–D burst with area b_1 x b_2 if $b_1' \leq b_1$ and $b_2' \leq b_2$.

Starting with the notion of a 2–D cyclic burst, following two special types of burst–error patterns are obtained by appropriate modifications. Both these burst–errors are correctable using spectral decoding methods as will be shown later.

2–D Quasi–cyclic burst with area t_1 x t_2 : Burst error e consists of an array $\{e_{i,j}\}$ of nonzero positions, such that $i \in \{i_1, i_2, ..., i_{t_1}\}$ and $j \in \{j_1, j_2, ..., j_{t_2}\}$.

Blahut's 2–D burst with area t_1 x t_2 : [2]–[4] A row–restricted Blahut's burst error consists of an array $\{e_{i,j}\}$ of nonzero positions having t_1 rows and each row containing not more than t_2 errors. A column–restricted Blahut's burst error is defined as an array $\{e_{i,j}\}$ of nonzero positions having t_2 columns and each column containing not more than t_2 errors. In general, a t_1 x t_2 row–restricted burst is different

from a t_1 x t_2 column—restricted burst. However, For the sake of convenience, both these bursts are referred to as Blahut's 2—D burst. The particular type of burst referred to is either evident from the context or made clear if necessary.

The above two 2—D burst error patterns are inter—related as follows: quasi—cyclic burst includes 2—D cyclic burst as a special case and quasi—cyclic burst is in turn a special case of Blahut's 2—D burst. A quasi—cyclic burst can be obtained by modifying an appropriate 2—D cyclic burst as follows: first spread a 2—D cyclic burst along the rows treating each nonzero rows of the 2—D burst as a unit, then do likewise along the column direction. As for Blahut's burst, the only difference is that after spreading once along the row or column direction, each row or column is further spread individually along the column or rows respectively. Figure 4.1.1 shows a typical 2—D cyclic burst pattern and also some of Blahut's burst pattern and quasi—cyclic burst pattern that can be obtained from it. The partial order relation w.r.t. these modified 2—D cyclic bursts is defined similarly as in the case of 2—D cyclic bursts.

a) 2 x 2 2—D cyclic burst

b) 2 x 2 quasi-cyclic burst

c) 2 x 2 Blahut's burst

Figure 4.1.1 Different types of 2—D burst—errors

X denotes error position

An interesting feature of spectral decoding algorithms is that it is not necessary to know about the state of the channel prior to decoding, i.e., whether the channel produces a particular type of burst error or

random errors only. This is unlike usual decoding algorithms which make an assumption that the channel produces either random errors only or burst–errors only. However, compared to decoding algorithms for concatenated codes which also have this feature [83]–[84], spectral decoding algorithms can be thought of as correcting similar types of complicated errors, but with a limited density only, that a concatenated decoding algorithm can handle. Hence these different burst errors can also be thought of as multiple low–density type of 2–D burst–errors as in [80].

4.2 Review of Blahut's decoding algorithms for random error–correction in 2–D BCH codes

Definition 4.2.1: [2]–[4] A 2–D BCH code is defined as the set of arrays $\{c_{i,i'}\}$ whose 2–D DFT (for $n = n_1 = n_2$) is given by

$$C_{a(m_j+j'),a'(m_j+j')+j} = 0 \; ; \; j=1,\ldots,2t; \; j'=1,\ldots,2t.$$

The minimum distance of 2–D code is at least 2t+1 provided (a,n) = 1.
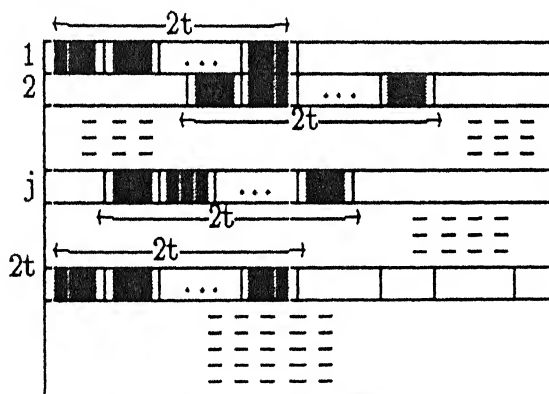
Figure 4.2.1 Zero locations of a 2–D BCH code

Shaded regions represent zero locations

Let us summarize the important steps in BA–1 for random error correction in 2–D BCH codes. Given the zero locations of 2–D BCH code in the form of a 2t x 2t contiguous array, first perform the B–M algorithm on each of the 2t rows (columns) to find the connection polynomial and use it to find the remaining syndromes in that row (column) by recursive extension. Then the process is repeated in the other direction for all of n columns (rows), i.e., performing B–M algorithm followed by recursive extension and the rest of the syndromes are found.

The recursive extension plays an important role in the development of decoding algorithms. The following theorem gives justification for Blahut's decoding algorithm for 2–D BCH codes. It is also stated

for different directions (horizontal, vertical or at any angle) in which the recursive extension can be carried out [4].

Theorem 4.2.1: Suppose $v \leq t$ random errors occur, and for any integers a, a' and j, j' the syndromes $S_{m+ak,m'+a'k}$ for $k=1,2,...,2t$ are known for a 2-D code of length $n_1 \times n_2$. Then these uniquely define the syndromes $S_{m+ak,m'+a'k}$ for $k=1, 2...n$, where $n = LCM(n_1,n_2)$.

Remark 4.2.1: Appropriate choice of a and a' determines both the direction of recursive extension and the period of the linear recurring sequence thus generated. It may be noted that the period of a syndrome sequence may be less than LCM (n_1,n_2). For instance, $a = 0$ corresponds to the case of recursive extension along a row, and the sequence thus generated will be periodic of length n_2, provided $(a',n_2)=1$. Similarly, the choice $a' = 0$ corresponds to the case of recursive extension along a column, and the sequence thus generated will be periodic of length n_1, provided $(n_1,a)=1$.

The following theorem due to Blahut [4] gives random error-correction capability of BA-1 for a $n \times n$ size 2-D BCH code having zero locations in the form of a $2t \times 2t$ contiguous array.

Theorem 4.2.2: Given any a, a', and m_k for $k = 1, 2,..., 2t$, the set of syndromes $S_{a(m_k+k'),a'(m_k+k')+k'}$ for $k= 1, 2,...2t$ and $k' = 1, 2, ..., 2t$, uniquely determines the error pattern, provided $v \leq t$ errors took place and $(a,n)=1$.

Remark 4.2.2: The roots of connection polynomial have interesting interpretation for recursion extension along row or column directions. If $e(x,y) = \sum_{k=1}^{v} y^{j_k} x^{i_k}$ is the error, then the error syndrome

$$S_{m+ak,m'+a'k} = \sum_{k=1}^{v} X_k^m Y_k^{m'} (Y_k^{a'} X_k^a)^k = \sum_{k=1}^{v'} Z_k (Y_k^{a'} X_k^a)^k$$

where $Y_k = \beta^{j_k}$, $X_k = \alpha^{i_k}$, and v' is the number of distinct $(Y_k^{a'} X_k^a)^k$ for $k = 1$ to v. According to Theorem 2.5.2 (Chapter 2), the initial conditions for the linear recurring sequence $\{S_{m+ak,m'+a'k}\}_{k=1}^{2t}$ are given by $Z_k \neq 0$ for $k = 1$ to v'. If $a = 0$ and $a' = 1$, the recursion is along the row-direction. The roots of the connection polynomial in this case are given by β^{j_k}, $k = 1$ to v, for which $Z_k \neq 0$. This means that if β^j is a root of the row connection polynomial, then i^{th} column is corrupted by the error (Note that if B-M algorithm of [4] is used for the computation of the connection polynomial, then the connection polynomial is produced in a reciprocal fashion such that if β^j is a root of the row connection polynomial, then $(n-i)^{th}$ column is corrupted by the error). Because some Z_k's may be zero, it may happen that the roots of the

connection polynomial may not give all the columns corrupted by the error. Similar interpretation holds for the roots of the column connection polynomial.

Remark 4.2.3: The following relations hold between connection polynomials corresponding to two different rows (columns) on the basis of Remark 4.2.2. For two rows, say, m^{th} row and m'^{th} row, where m and m' are conjugate of each other w.r.t. n, both $e_k(\alpha^m)$ and $e_k(\alpha^{m'})$ are either simultaneously zero or nonzero, and therefore both row sequences satisfy the same row connection polynomial.

Remark 4.2.4: Let $e(x,y) = \sum_{k=1}^{v} y^{j_k} x^{i_k}$ be the error and assume that j_k's are all distinct. If $\Lambda_m(z)$ and $\Lambda_n(z)$ are the row connection polynomials for m^{th} row and n^{th} row respectively, then it is known that the least common multiple of Λ_m and Λ_n satisfies row syndromes of both the rows m and n [74]. It is of practical interest to find the common row connection polynomials as it may reduce book keeping. The following lemma shows how a simple modification of B–M algorithm produces the required connection polynomial.

Lemma 4.2.1: If $\Lambda_m(z)$, the common row connection polynomial satisfied by the first m rows, is known then $\Lambda_{m+1}(z)$ is obtained by passing row syndromes of $m+1^{th}$ row through the B–M algorithm with the initialization $\Lambda_{m+1}^{(0)}(z) = B_{m+1}^{(0)}(z) = \Lambda_m(z)$.

Proof: $\Lambda_m(z)$ can be considered as "virtual erasure polynomial" for the $m+1^{th}$ row syndromes as it gives partial information on the columns affected by the error. Then proof follows from similar initialization of the B–M algorithm with the erasure polynomial for the case of errors–and–erasures decoding [4]. Therefore for $m+1^{th}$ row syndromes the iterations of B–M algorithm starts at $\rho+1$, where ρ is the degree of $\Lambda_m(z)$, and when 2t iterations of the B–M algorithm are completed $\Lambda_{m+1}(z)$ is obtained. □

Remark 4.2.5 : The Lemma 4.2.4 gives a method of finding the common connection polynomial and it is important to know the minimum number of passes through B–M decoder required to find the common connection polynomial. The claim is that at most 't' passes through B–M algorithm corresponding to 't' consecutive rows will do. The reason is as follows. Consider a particular instance of a random error $e(x,y) = g_1(x)y^j$ affecting j^{th} column. If $g_1(\alpha^j) = 0$, for j = 1 to t, implies $g_1(x)$ to be of weight greater than t contradicting the hypothesis. Therefore $g_1(\alpha^j) \neq 0$, for at least one j ϵ {1,2,...,t}, and the corresponding column error locator is among the roots of the common row connection polynomial. It is clear that the procedure described above requires at most half the number of passes through B–M decoder compared to the original decoding algorithm of Blahut and therefore gives an improvement of the speed of decoding.

The same result is proved in a different way by Fitzpatrick et.al. [51] for the general case of m–dimensional BCH codes. However, our approach is different and the arguments can be easily extended for higher dimensions.

Remark 4.2.6: There is another simple criterion that reduces the number of passes through the B–M algorithm sometimes. If it happens for some row or column that a pass through B–M algorithm gives a connection polynomial of degree 't', then the same row or column connection polynomial can be used for recursive extension of subsequent rows or columns without need for a pass through the B–M algorithm. Therefore some computational advantages can be obtained over BA–1.

Example 4.2.1: Consider the following 2–D BCH code consisting of arrays of size 5 x 5. The zeros of the code are $C_{j,j'} = 0$, for j=1 to 4 and j'=1 to 4 (j is along x–direction and j' is along y–direction). In Figure 4.2.2 the locations within the bordered area denote available syndromes. Let $e(x,y) = xy + (y+y^4)x^3$ be the error pattern, which is an example for quasi–cyclic burst with area 2 x 2. (x^4+x+1) is the minimum polynomial for generating $GF(2^4)$, with α as the primitive element.

$$
\begin{array}{c}
\\
\\
S =
\end{array}
\begin{array}{c}
\\
1 \\
2 \\
3 \\
4 \\
0
\end{array}
\begin{array}{cccc|c}
1 & 2 & 3 & 4 & 0 \\
\hline
\alpha^{12} & \alpha^4 & \alpha^5 & \alpha & \alpha^3 \\
\alpha^{10} & \alpha^9 & \alpha^2 & \alpha^8 & \alpha^6 \\
\alpha^2 & \alpha^8 & \alpha^6 & \alpha^{10} & \alpha^9 \\
\alpha^4 & \alpha^5 & \alpha & \alpha^3 & \alpha^{12} \\
\hline
\alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1
\end{array}
$$

Figure 4.2.2 Recursively extended complete syndrome array

It follows from Remark 4.2.3 that it is sufficient to find the connection polynomial for row 1, which is found to be $(1 + \alpha^{10}z + z^2)$. Recursively extending along the first four rows find $S_{1,0}$, $S_{2,0}$, $S_{3,0}$ and $S_{4,0}$. Similarly it is enough to find the connection polynomial for column 1 which is found to be $(1 + \alpha z + \alpha^{12}z^2)$, and recursion along the column direction produces all the remaining syndromes. A 2–D IDFT of S gives e.

4.3 Blahut's decoding algorithms for burst–error correction

We now consider correction of quasi–cyclic and Blahut's 2–D burst–errors introduced earlier. We first show that BA–1 is capable of inherently correcting quasi–cyclic 2–D burst errors. We then suggest some modifications to BA–2 following modifications to BA–1 for efficient implementation.

To begin with, we summarize important steps in Blahut's decoding algorithm for burst–error correction (referred to as BA–2). The nature of 2–D burst–errors to be corrected by BA–2 is such that

either at most 't' rows or at most 't' columns are corrupted and further each corrupted row or column contains not more than t errors. The decoding algorithm starts by performing the B–M algorithm on each of the 2t rows. Since it is possible for a row to have more than t errors depending upon the burst–error pattern to be corrected, the connection polynomial has to checked to ascertain its validity. The periodicity test on the generated row syndrome is sufficient to ensure that the number of the roots of the connection polynomial is equal to its degree and that each of the roots correspond to valid error locator [4]. If the result is not a legitimate connection polynomial then perform the B–M algorithm on each of 2t columns to find the legitimate column connection polynomial. In any case, use the legitimate connection polynomial to find the remaining syndromes in the first 2t rows or columns by recursive extension. Next, take 1–D IDFT of these 2t rows or columns and identify the corrupted rows or columns. Now there are at most t such rows or columns each containing not more than t errors. Further for each corrupted row or column 2t consecutive syndromes of the row or column error–pattern are known, which can recover column or row error–pattern using the BCH decoding algorithm.

It is found convenient to introduce a particular type of 2–D BCH code having a simple zero structure, which simplifies proofs. Even in choosing the direction of the syndrome extension, it helps to choose the row recursion and and the column recursion, as these choice help in easily visualizing how the decoding algorithm is proceeding. Such a 2–D BCH code has a zero spectral window of size 2t x 2t and the parameters of the code are $a = 1$, $a' = 0$, $m_k = 0$ for $k = 1, 2,..., 2t$. Then the zero locations are given by (See Definition 4.2.1)

$$C_{k',k} = 0; \quad k = 1..2t; \quad k'=1..2t; \tag{4.3.1}$$

Henceforth, this choice of zero spectral locations is assumed, unless otherwise stated. Later on, it is shown that the decoding of any 2–D BCH code is equivalent to the decoding of a 2–D BCH code of the above type.

In order to show that BA–1 can also correct burst error patterns, a generalization of Theorem 4.2.1, assuming burst–errors, is needed. It is this idea which is crucial for proving that the BA–1 can also correct all 2–D quasi–cyclic burst–errors with area less than or equal to t x t. We have the following lemma.

Lemma 4.3.1: Suppose $e(x,y) = \sum_{k=1}^{v} y^{j_k} e_k(x)$ be the error pattern, where $v \leq t$ and j_k's are distinct; and for any integer m the syndromes $S_{m,m'}$ for $m' = 1, 2, ..., 2t$ are known. Then these uniquely define the syndromes $S_{m,m'}$ for $m'=1, 2,..., n$.

Proof: Proof runs almost parallel to the proof for the case of random errors given in [3]–[4]. For the given $e(x,y)$, the error syndrome can be written as

$$S_{m,m'} = \sum_{k=1}^{v} e_k(\alpha^m)\beta^{j_k m'} \qquad (4.3.1a)$$

or

$$S_{m,m'} = \sum_{k=1}^{v} Z_k(Y_k)^{m'} \qquad (4.3.1b)$$

where $Y_k = \beta^{j_k}$ and Z_k denotes the factor multiplying Y_k's, for k = 1 to v. Since Z_k's does not depend on m', the index of recursion, and there are $v \leq t$ distinct terms in (4.3.1b), the B–M algorithm on $S_{m,m'}$ for m'=1 to 2t, produces the legitimate connection polynomial of the sequence. The remaining syndromes for m^{th} row are determined from recursive extension. □

Theorem 4.3.1: For a 2–D BCH code with zero positions given by $C_{k',k}=0$, for k=1 to 2t and k'=1 to 2t, BA–1 can correct any quasi–cyclic 2–D burst–error with area less than or equal to t x t.

Proof : A quasi–cyclic 2–D burst with area less than or equal to t x t can be written as

$$\sum_{k=1}^{v} y^{j_k} e_k(x) = \sum_{k=1}^{v'} x^{i_k} e_k'(y)$$

where $1 \leq v,v' \leq t$ and j_k's and i_k's are distinct. Therefore, for each k'=1 to 2t, the sequence $\{S_{k',k}\}_{k=1}^{2t}$, by Lemma 4.3.1, determines the sequence $\{S_{k',k}\}_{k=1}^{n}$. Thus, after processing 2t rows, $S_{k',k}$ for k' = 1 to 2t and k = 1 to n, are known. Now invoking Lemma 4.3.1 for each of the sequence $\{S_{k',k}\}_{k'=1}^{2t}$, for k = 1 to n, the remaining syndromes are found. □

Remark 4.3.1: For the case of 2–D BCH code given in Definition 4.2.1, the decoding proceeds as follows. The first step is to consider the automorphism of the abelian group given by $x' \longrightarrow x^a y^{a'}$ and $y' \longrightarrow y$ (This mapping represents automorphism mapping since (a,n) = 1. [85]) in the spectral domain, which modifies the zero spectral locations of the general 2–D BCH code as:

$$C_{m_k+k',k} = 0; \text{ for } k=1..2t; k'=1..2t;$$

Then the correction of quasi–cyclic errors with these spectral locations is similar to the code with zero locations as in (4.3.1) except for the following modification: first perform the B–M algorithm on 2t columns followed by recursive extension in the column direction and find all the syndromes in those 2t columns. Then repeat the procedure n times in the row direction to determine all the remaining syndromes.

We now consider modifications to Blahut's algorithm for correction of Blahut's 2–D burst–errors. Since BA–2 also involves computation of the common row or column connection polynomial, respective modifications suggested for BA–1 are carried over to this algorithm. Another suggestion for modification is to make use of roots of the row (column) connection polynomials to determine the columns (rows) which

are affected by error. The idea of the following theorem implicit in Blahut's work, is stated for the sake of completeness.

Theorem 4.3.2: For a 2–D BCH code with the syndromes $S_{j,j'}$ for $j = 1$ to $2t$ and $j' = 1$ to $2t$, BA–2 can correct Blahut's 2–D burst–error with area less than or equal to $t \times t$.

Proof : The burst error affects either $v \leq t$ columns or $v \leq t$ rows. Assume wlg that $v \leq t$ rows are affected and therefore the common column connection polynomial can be computed. The distinct roots of this common column connection polynomial identify the locations of $v \leq t$ corrupted rows.

We now be show that 1–D IDFT of the $2t$ known columns of the syndrome window, at row error positions, produces $2t$ consecutive sub–syndromes in those rows, from which individual row error–pattern having $v \leq t$ errors can be recovered. We have for j^{th} column of S

$$S_{j,j'} = E_{j,j'} = \sum_{i=0}^{n-1} \left[\sum_{i'=0}^{n-1} e_{i,i'} \beta^{i'j'} \right] \alpha^{ij} = \sum_{i=0}^{n-1} E'_{i,j'} \alpha^{ij} .$$

Then 1–D IDFT of the jth column syndromes $\{S_{j,j'}\}_{j=0}^{n-1}$, gives $\{E'_{i,j'}\}_{i=0}^{n-1}$, where $E'_{i,j'}$ is the jth coefficient of 1–D DFT of $\underline{e}_i^{(r)}$, the ith row of $\{e_{i,i'}\}_{i'=0}^{n-1}$. Hence $S'_{i,j'}$ for $j' = 1,...,2t$, gives $2t$ consecutive syndromes for i^{th} row error pattern. Using B–M algorithm we can recover $\underline{e}_i^{(r)}$ of e provided $v \leq t$ errors have occurred in that row.

The proof is similar for the case of 1–D IDFT of the $2t$ known rows of the syndrome window. Then the $2t$ consecutive column sub–syndromes are known from which column error pattern is determined provided $v' \leq t$ errors have occurred in that column.   □

Remark 4.3.2: As for the correction of Blahut's 2–D burst in the general case, the same procedure given in Remark 4.3.1 holds good. Then it is only possible to correct column–restricted Blahut's burst–error. This because of the constraint of having to process first the column syndromes to find the connection polynomial.

4.4 Improved Blahut's decoding algorithms

We now present improved Blahut's algorithms incorporating the modifications suggested above. The first algorithm is stated for simultaneous correction of random errors and quasi–cyclic 2–D burst–errors upto area $t \times t$. The second algorithm is stated for correction of column–restricted Blahut's 2–D burst–error correction. The modifications for correction of row–restricted Blahut's burst is straight forward. For some 2–D BCH codes which can correct both row–restricted and column restricted bursts, it is necessary to include validity tests for connection polynomial as explained earlier.

Improved Blahut's Decoding Algorithm–I (IBA–I)

1) Obtain 2–D DFT of r to obtain R.

Set $S_{m_k+k',k} = R_{a(m_k+k'),a'(m_k+k')+k'}$ for $k = 1$ to $2t$ and $k' = 1$ to $2t$.

2) For (columns) $k = 1$ to t do

if $(\deg(\Gamma^k) \neq t$ and $(1 \leq k \leq t))$, then

Set $S'_{k'} = S_{m_k+k',k}$ for $k'=1$ to $2t$;

Perform B–M algorithm on $\{S'_{k'}\}_{k'=1}^{2t}$ with the initialization $\Gamma^k_{(0)} = B^k_{(0)} = \Gamma^{k-1}$ and compute Γ^k. Set $\Gamma^c = \Gamma^k$

3) For (columns) $k = 1$ to $2t$ do

For $k' = 2t+1$ to n, compute $S'_{k'} = \sum_{j=0}^{n-1} \Gamma^c_j S'_{(k'-j)}$

Set $S_{(m_k+k')_n,k} = S'_{k'}$

4) For (rows) $k' = 1$ to t do

if $(\deg(\Gamma^{k'}) \neq t$ and $(1 \leq k' \leq t))$, then

Set $S'_k = S_{k',k}$ for $k = 1$ to $2t$;

Perform B–M algorithm on $\{S'_k\}_{k=1}^{2t}$ with the initialization $\Gamma^{k'}_{(0)} = B^{k'}_{(0)} = \Gamma^{k'-1}$

and compute Γ^r

5) For (rows) $k' = 1$ to n do

For $k = 2t+1$ to n, compute $S'_k = \sum_{j=0}^{n-1} \Gamma^r_j S_{(k-j)}$

Assign $S_{k',k} = S'_k$

6) For $k = 1$ to n, $k' = 1$ to n, set

$S''_{a(m_k+k'),a'(m_k+k')+k} = S_{m_k+k',k}$

7) For $i = 1$ to n and $j = 1$ to n, set

$C_{i,j} = R_{i,j} - S''_{i,j}$

8) Compute 2–D IDFT of C to obtain c

Improved Blahut's Decoding Algorithm–II (IBA–II)

1) Compute R and obtain column connection polynomial Γ^c as in IBA.

2) Find the roots of Γ^c and determine v rows, i_1 to i_v, that are in error.

3)   For k = 1 to 2t do

Compute $\underline{E}_k^{!(c)}$, 1–D IDFT of $\{S_{k',k}\}_{k'=0}^{n-1}$, and obtain $E_{i_j,k}^!$ for j=1 to v.

4)   For j = 1 to v do

Use B–M algorithm on $\{E_{i_j,k}^!\}_{k=1}^{2t}$ to find the connection polynomial

and recursively extend to $\{E_{i_j,(k)_n}^!\}_{k=1}^{n}$.

Compute 1–D IDFT of $\{E_{i_j,k}^!\}_{k=0}^{n-1}$ to obtain i_j^{th} row of e.

5)   The code vector is recovered as c = r ⊕ e.

4.5 Burst–error correction capability of 2–D BCH codes

In the previous section the correction capability of Blahut's decoding procedure is studied for the case of both random and burst errors. However, correcting capability of the code as such, whatever may be the decoding method employed, is also important. In what follows we employ 2–D complexity theorem discussed in Chapter 2 to determine the correction capability of a 2–D BCH code with regard to both quasi–cyclic and Blahut's 2–D burst–errors. It is shown that, for both of these types of bursts, Blahut's decoding algorithms are best possible in that correction upto the error–correction limit is possible [87].

A code A is said to have 2–D cyclic burst–error correcting capability upto area t x t, if it corrects every 2–D cyclic burst–error with area less than or equal to t x t and does not uniquely decode every 2–D burst–errors with area equal to t+1 x t+1. Such a code A has unique syndrome for each 2–D cyclic burst–error with area less than or equal to t x t. The notions of quasi–cyclic and Blahut's 2–D burst–error correcting capabilities of a code upto area t x t are defined similarly.

The following theorem concerns quasi–cyclic burst error–correcting capability of 2–D BCH codes.

Theorem 4.5.1: A 2–D BCH code with the zeros $C_{j,j'}= 0$, for j=1 to 2t and j'=1 to 2t, has quasi–cyclic burst–error correcting capability t x t.

Proof: The proof is by contradiction. Suppose two distinct quasi bursts b_1 and b_2 with area less than or equal to t x t, have the same syndrome pattern in the 2t x 2t spectral window. Then B_1-B_2, which is the 2–D DFT of b_1-b_2, has all zeros in the zero spectral window of area 2t x 2t. From the complexity theorem, it follows that the minimal degree connection polynomial for any row or column syndrome sequence is at most 2t, i.e., maximum possible number of columns that can be corrupted. As B_1-B_2 is all zero in the 2t x 2t spectral window, after recursive extension B_1-B_2 is all zero vector, and therefore $B_1 = B_2$ which implies $b_1 = b_2$. This contradicts the assumption that b_1 and b_2 are distinct. Therefore

quasi–cyclic burst–correcting capability is at least t x t. It is well known from LFSR theory that 2t continuous syndromes may have the same spectral pattern for two distinct errors of weight t+1 and considering two such 1 x (t+1) tuples, it is easy to see the non–uniqueness of syndromes. Hence the theorem follows.

□

The error correcting capability of 2–D BCH code in respect of Blahut's burst–error is given in the following theorem.

Theorem 4.5.2: A two–dimensional BCH code with the zeros $C_{j,j'} = 0$, for j = 1 to 2t and j' = 1 to 2t, has Blahut's burst–error correcting capability t x t [87].

Proof: Suppose b_1 and b_2, two Blahut's burst–error with area less than or equal to t x t, have the same syndrome pattern in the 2t x 2t spectral window. W.l.g, assume that row–restricted b_1 contains $v_1 \leq t$ nonzero rows. There are two cases to be considered and Figure 4.5.1 shows typical burst error patterns for both cases. For simplicity it is assumed that burst errors occur in continuous rows or columns.
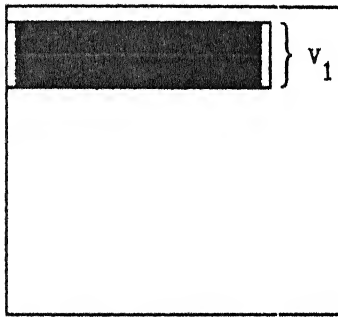
a) Burst b_1
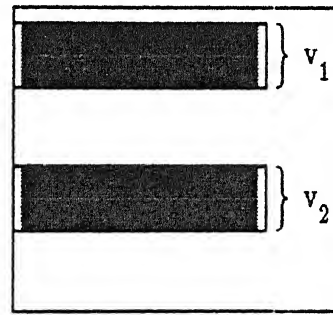
b) Bursts b_1 & b_2 (Case a)
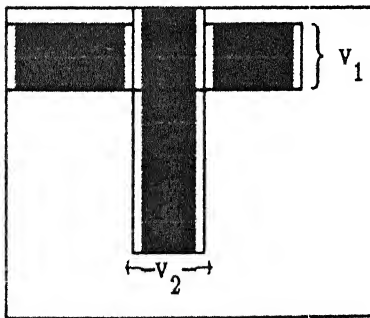
c) Case (b)–(i)
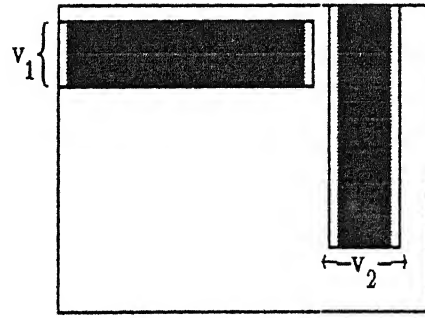
d) Case (b)–(ii)

Figure 4.5.1 Different types of 2–D burst–error configurations

Case (a): b_2 contains $v_2 \leq t$ corrupt rows (Fig. 4.5.1b):

By 2—D complexity theorem, the degree of any column connection polynomial is at most 2t, and this implies that $B_1 - B_2$ will have all zeros in the first 2t continuous columns. This further implies that each nonzero row of $b_1 - b_2$ should have weight at least 2t+1. However, even if b_1 and b_2 overlap there are no more than 2t nonzero positions in any nonzero row. This contradiction shows that b_1 and b_2 must have different spectral patterns.

Case (b): b_2 contains $v_2 \leq t$ corrupt columns (Figures 4.5.1 c and d)

The degree of row connection polynomial of B_2 and degree of column connection polynomial of B_1 are both at most t. By hypothesis, $B_1 = B_2$. Therefore, the decoding algorithm outputs an error burst b' whose rows are those of b_1 and columns are those of b_2. This implies that both $b_1 - b'$ and $b_2 - b'$ are valid code words of the 2—D BCH code. However, either of them is not a code word as shown in case (a). Hence the contradiction obtains.

Since a quasi—cyclic 2—D burst is a special case of a Blahut's 2—D burst, it is already shown in Theorem 4.5.1 that it is not possible to correct 2—D bursts with area 1 x t+1. □

Remark 4.5.1: It follows that for the 2—D BCH code with the defining zeros $C_{j,j'} = 0$ for $1 \leq j \leq t$ and $1 \leq j'$ $\leq t$, that BA—1 is the best possible as for as the correction of quasi—cyclic burst—error correction and BA—2 is the best possible for Blahut's 2—D burst error—correction. This fact also holds true for the case of general 2—D BCH codes, although it is not possible to utilize full capability of BA—2 as observed in Remark 4.3.2.

4.6 A new class of Γ 2—D BCH codes and their decoding

It is shown in [81] that a linear code correcting 2—D cyclic burst of area upto t x t requires minimum of $2t^2$ parity checks. The class of 2—D BCH codes defined above requires $4t^2$ bits of parity check which is twice the bound of $2t^2$. As an effort towards reducing the excess parity checks, a new class of codes, namely, Γ 2—D BCH codes having $3t^2$ parity checks is introduced. It will be shown that these Γ 2—D BCH codes have the same error—correcting capability as that of 2—D BCH codes with defining zero locations in the form of a 2t x 2t contiguous array. Therefore Γ 2—D BCH codes have t^2 more parity checks than the class of Γ 2—D BCH codes for the same error—correcting capability. It is to be noted that BA—1 and BA—2 are not directly applicable to decode the class of Γ 2—D BCH codes, but IBA can be readily applied to decode upto the error—correcting capability.

Definition 4.5.1: A Γ 2–D BCH code is defined as the set of n x n arrays c whose 2–D DFT satisfies

$$C_{a(m_j+j'),a'(m_j+j')+j} = 0; \quad (j, j') \in S_1/S_2$$

where $S_1 = \{(j,j')|j = 1..2t, j' = 1..2t\}$ and $S_2 = \{(j,j')| j = t+1..2t, j' = t+1..2t\}$ and '/' is the set difference operator and $(a,n) = 1$.
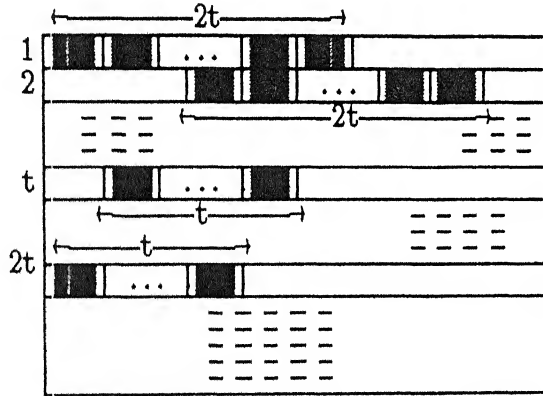
Figure 4.6.1 Zero locations of a typical Γ 2–D BCH code
Shaded regions represent zero locations

To investigate error–correcting capabilities choice of zero locations having simple structure corresponds to $m_j = a' = 0$ and $a=1$. Then the zero locations are given by:

$$C_{j'j} = 0 ; \quad (j,j') \in S_1/S_2 \tag{4.6.1}$$

where $S_1 = \{(j,j')| j=1..2t, j'=1..2t\}$ and $S_2 = \{(j,j')| j = t+1..2t, j' = t+1..2t\}$. Error–correcting capability of this class of Γ 2–D BCH code with respect to Blahut's burst–error correction is given in the following theorem.

Theorem 4.6.1: A Γ 2–D BCH with the zero locations defined as in Equation (4.6.1) has t x t Blahut's 2–D burst–error correction capability.

Proof: Suppose b_1 and b_2, two Blahut's burst–error with area less than or equal to t x t, have the same syndrome. W.l.g. assume that b_1 contains $v_1 \leq t$ corrupt rows. There are two cases to be considered, as in the proof of Theorem 4.5.2 (see Figure 4.5.1). Only Case (a) requires a new proof. The proof of Case (b) is identical and therefore omitted.

Case (a): b_2 contains $v_2 \leq t$ corrupt rows (Figure 4.5.1a)

It follows from 2–D complexity theorem that the degree of any column connection polynomial is at most 2t, and this implies that B_1–B_2 will have all zeros in t consecutive columns. Then it follows that each nonzero row of b_1–b_2 should have weight at least t+1. If the total number of nonzero rows in b_1–b_2 exceed t, then there is at least one nonzero row having t nonzero positions and therefore contradiction obtains. On the other hand, if the total number of nonzero rows is less than or equal to t, from the defining zero structure of the code, it follows that B_1–B_2 will have all zeros in 2t consecutive columns. This further implies that any nonzero row should have weight of at least 2t+1. But there are no more than 2t nonzero positions in any nonzero row and contradiction is established. Therefore b_1 and b_2 must have different spectral patterns.

Case (b): b_2 contains $v_2 \leq t$ corrupt columns (Figures 4.5.1 c and d)

The proof is same as in Theorem 4.5.2. □

Remark 4.6.1: It is shown in Remark 4.2.5 that given t consecutive row (column) syndromes, IBA takes at most t passes through B–M algorithm to find the common row or column connection polynomial and therefore applicable for random–error and burst–error correction in the case of Γ 2–D BCH with simple modifications. It also follows from Theorem 4.6.1 that the IBA is the best possible decoding algorithm for the class of Γ 2–D BCH codes.

Remark 4.6.2: It can be shown along the same lines as in Remark 4.3.1, that the decoding of the class of Γ 2–D BCH code for the general case is equivalent to the decoding of a Γ 2–D BCH code of the above type. The automorphism of the abelian group given by $x' \rightarrow x^a y^{a'}$ and $y' \rightarrow y$ in the spectral domain, which modifies the zero spectral locations of the general Γ 2–D BCH code as:

$$C'_{m_k+k',k} = 0; \text{ for } (k, k') \in S_1/S_2$$

The decoding with these spectral locations is then similar to the case with $m_k =0$, except for the following modification: first perform B–M algorithm on t columns followed by recursive extension in the column direction and find all the syndromes in those 2t columns. Then repeat the procedure n times in the row direction to determine all the remaining syndromes.

Example 4.6.1: Consider the following Γ 2–D BCH code consisting of arrays of size 7 x 7 taking values from $GF(2^3)$. The locations within the bordered area of Figure 4.6.2 represent syndromes, $S_{j,j'}$ for j=1 to 3, j'=1 to 6 and for j=1 to 6, j'=1 to 3, corresponding to zero locations. Let $e(x,y) = y(1+\alpha x+a^4 x^2) + y^2(1+x+a^2 x^2) + y^3(1+a^6 x+x^2)$ be the error pattern, which is an example of 3 x 3 quasi–cyclic 2–D burst–error. (x^3+x+1) is the minimum polynomial generating $GF(2^3)$, with α as the primitive element.

	1	2	3	4	5	6	0
1	a^6	1	1	0	a^5	a^3	1
2	a^2	a^5	a	a^4	1	a^3	a^6
3	a^3	a^4	a^5	a^6	1	a	a^2
4	a^2	1	1	a^6	0	a^2	a^6
5	a	a^4	a^5	0	a^4	a^6	0
6	a	0	a^2	a^3	a^5	a	0
0	a^3	0	a^2	a^4	0	a^6	a^2

Figure 4.6.2 Recursively extended syndrome array

It is clear that BA–1 & BA–2 cannot be directly used. Even if it were possible to decode, algorithm will be complex. Fortunately, IBA can be applied with little modifications. Start decoding with row 1 syndromes and the connection polynomial is found to be $(1 + a^5 z + a^5 z^2)$. Syndromes in row 2 also satisfy the same connection polynomial. However, B–M algorithm for row 3 syndromes initialized with this connection polynomial gives the common row connection polynomial as $(1 + a^6 z + az^2 + a^6 z^3)$. As rows 4 to 6 have 3 consecutive syndromes, performing recursive extension along the row direction all the syndromes in first 6 rows are known. Processing column 1 using B–M algorithm gives the connection polynomial as $(1 + a^5 z + a^6 z^2 + a^3 z^3)$. Since this is of degree 3, no more further passes through B–M algorithm is required and recursive extension followed by 2–D IDFT completes the decoding.

Remark 4.6.3: Another interesting question is whether it is possible to reduce further the number of zeros i.e. $3t^2$ in the definition of a Γ 2–D BCH code in order to correct 2–D cyclic burst error with area $\leq$ t x t. Answer to this question is negative as can be shown below. Consider the code A' with the following zero locations:

$$C'_{j,j'} = 0; \quad (j, j') \in S_0 / S_1$$

where $S_0 = \{(j,j') \mid 1 \leq j \leq 2t, \ 1 \leq j' \leq 2t\}$ and $S_1 = \{(j,j') \mid t+1 \leq j \leq 2t, \ t+1 \leq j' \leq 2t\} \cup \{(1,2t)\}$. The zero locations are the same as that of Γ 2–D BCH code except that one zero location is dropped (W.l.g. it is assumed (1, 2t) chosen as the information location. The proof is similar if any other zero location is chosen as information location). It is easy to verify that the following $c(x,y) = \prod_{k=2}^{t} (1 - xa^{-k}) \prod_{h=1}^{2t-1} (1 - y\beta^{-h})$ is a code word. But $c(x,y)$ can be expressed as sum of two 2–D cyclic bursts with area t x t. This shows that the code A' defined above cannot be a t x t cyclic burst–error correcting code.

4.7 Mixed spectral and time domain implementation of Blahut's decoding algorithms

We have so far discussed decoding of 2–D BCH codes based on spectral domain implementation of B–M algorithm. This means that the recursive computations of B–M algorithm operate on the error syndromes which are to obtained from as the 2–D DFT of the received vector. Blahut has shown that for 1–D BCH decoding it is possible to implement the B–M algorithm in the time domain and given a decoding algorithm that operates directly on the time domain received data [4]. The main advantage of time domain processing for 1–D BCH codes lies in eliminating computations of forward and inverse DFT required in the case of spectral domain processing. There is no syndrome computer or Chien search and this leads to simpler decoder architectures. However, the number of major clocks required is equal to the code length (n) independent of errors corrected and further during each clock time n–length vectors have to be processed.

For the 2–D case, these ideas can be extended but with some modifications. This is because the concept of syndrome is 2–D and that of error–locator as applied to decoding 2–D BCH codes is shown to be essentially 1–D. It is well–known that a 2–D DFT computation can be seen as two stage computation consisting of 1–D DFT of all the rows followed by 1–D DFT of all the columns. Therefore by processing 1–D DFT of rows (columns) of the received vector, it becomes possible to use time domain B–M algorithm. For this reason, this method is called mixed spectral and time domain (MSTD) decoding algorithm. It is also clear that a 2–D decoding using MSTD approach cannot have the complete advantage of eliminating DFT computation.

We develop two important lemmas that together give MSTD decoding algorithm. For simplicity of exposition we take the zero locations of the 2–D BCH code to be

$$C_{j,j'} = 0; \text{ for } j=1,.., 2t \text{ and } j'=1,.., 2t \tag{4.7.1}$$

For Γ 2–D BCH code the zero locations are given by

$$C_{j,j'} = 0; \text{ for } j=1,.., 2t , j'=1,.., t$$
$$C_{j,j'} = 0; \text{ for } j=1, ., t, j'=t+1,.., 2t \tag{4.7.2}$$

We observe the notation that if r is any time–domain vector, then R' denotes the vector consisting of 1–D DFT of all columns of r and R denote 2–D DFT of r. $\underline{R}_j^{(r)}$ denotes j^{th} row of R and $\underline{R}_i^{(c)}$ denotes i^{th} column of R.

Lemma 4.7.1 : Suppose $r = c \oplus e$ be the transmitted code vector and $v \le t$ columns are in error. Then the time–domain B–M algorithm on the rows, 1 to 2t of R', produces rows, 1 to 2t of E'.

Proof: It follows from the definition of R' that

$$R_{j,j'} = \sum_{i'=0}^{n-1} R'_{j,i'} \beta^{i'j'}$$

For j^{th} row syndromes we have $E_{j,j'} = R_{j,j'}$ for $j'=1$ to $2t$, and it follows that error-locator vector Λ can be found using spectral domain B-M algorithm satisfying

$$\Lambda * \underline{E}_j^{(r)} = 0$$

Therefore time domain B-M algorithm processing $\underline{R}_j^{'(r)}$, j^{th} row of R', finds auxiliary vector λ satisfying

$$\lambda . \underline{E}_j^{'(r)} = 0. \qquad \qquad \square$$

Remark 4.7.1: We have interpretation that $\lambda_i = 0$ means i^{th} column is affected by the error. Also for rows m and m', whose indices are conjugates, time-domain B-M algorithm produces the same auxiliary polynomial. For a 2-D BCH code with the zeros in the form of a $2t \times 2t$ contiguous array, the common auxiliary vector is given as the pointwise product of λ's of corresponding to t consecutive rows (columns). The column errors are then given by the zero valued locations of the common auxiliary polynomial. If the rows are serially processed then by initializing $\lambda_{m+1}^{(0)}$ and $b_{m+1}^{(0)}$ of $m+1^{th}$ row by λ_m of mth row, the common auxiliary column vector $\lambda^{(c)}$ is obtained after processing of m consecutive rows. The proof is analogous to that of Lemma 4.2.1 and therefore omitted.

After $2t$ rows corresponding to the syndrome window of R' are processed, $v \leq t$ columns containing errors are identified and also $2t$ consecutive sub-syndromes in each corrupted column are known. Therefore passing these column sub-syndromes through spectral domain B-M decoder followed by IDFT produces the error pattern. The following lemma shows that the IDFT of appropriately zero-padded sub-syndrome column vector passed through time domain B-M decoder produces the column error pattern.

Lemma 4.7.2: Suppose $S = (S_0, S_1, S_2,..., S_{2t},..., S_{n-1})$ be the 1-D DFT of the column error pattern s_c. Consider another vector $S' = (0, S_1, S_2,..., S_{2t}, 0,..., 0)$ and s' be the IDFT of S'. Then time-domain B-M algorithm processing s' gives column error pattern s.

Proof. Let Ψ be the spectral domain connection polynomial produced by the sequence $S_1, S_2,..., S_{2t}$. The first step is to verify that first $2t$ recursions of time-domain B-M algorithm produces the time domain auxiliary polynomial ψ, the IDFT of Ψ. We have

$$\Delta_r = \sum_{i=0}^{n-1} \alpha^{ir} \psi_i^{(r-1)} s_i'$$

or

$$\Delta_r = \sum_{j=0}^{n-1} \Psi_j^{(r-1)} S_{r-j}'$$

But $S_j^! = S_j$, for j=1 to 2t, and $\Psi_j^{(r-1)} = 0$ for $j > 2t$. Therefore

$$\Delta_r = \sum_{j=0}^{n-1} \Psi_j^{(r-1)} S_{r-j} \, ,$$

and therefore time–domain B–M algorithm on $s^!$ produces the time domain auxiliary polynomial ψ.

We now show that next (n–2t) recursions produce column error s. For $r = 2t$ to n, it is easily verified that $\Delta_r = S_r$, and r^{th} component of $S^!$ changes to S_r. Hence the corresponding time–domain recursion produces the column error pattern s. □

We now proceed to give on MSTD decoding for the class of 2–D BCH and Γ 2–D BCH codes. The zero locations for these codes are given by (4.7.1) and (4.7.2) respectively. At step 2 of the decoding algorithm t rows are processed concurrently. If rows are to be serially processed then the initialization for each row is done as indicated in Remark 4.7.1. At step 4, Γ 2–D BCH codes are processed differently. This is because Γ 2–D BCH code has t zero locations in the rows t+1 to 2t, and therefore time–domain vector has to be updated (n–t) times.

Mixed Spectral and Time Domain Decoding Algorithm
for 2–D BCH and Γ 2–D BCH Code

1) Compute $R^!$ by taking 1–D DFT of all the columns of r.

2) For j=1 to t, pass $\{R^!_{j,k}\}_{k=0}^{n-1}$ through time–domain B–M algorithm to find $\lambda^{(j)}$ and $\underline{E}_j^{!(r)}$.

3) Compute the pointwise product $\lambda^{(c)} = \lambda^{(1)}\lambda^{(2)}...\lambda^{(t)}$ and identify the columns j_k for k=1 to v, for which $\lambda_{j_k}^{(c)} = 0$.

4) For $j = t+1$ to 2t, pass $\{R^!_{j,k}\}_{k=0}^{n-1}$ through time–domain B–M algorithm to find $\underline{E}_j^{!(r)}$. As $\lambda^{(c)}$ is known only the last (n–2t) iterations of time–domain B–M algorithm have to be done. For Γ 2–D BCH codes last (n–t) iterations have to be done.

5) For columns j_k, k = 1 to v, extend 2t syndromes $E^!_{i,j_k}$, for i= 1 to 2t, to $\underline{E}_{j_k}^s = (0, E^!_{1,j_k}, E^!_{2,j_k}, ..., E^!_{2t,j_k}, 0, ..., 0)$ and compute IDFT of $\underline{E}_{j_k}^s$ to obtain $\underline{e}_{j_k}^s$.

6) Pass $\underline{e}_{j_k}^s$, for k = 1 to v, through time–domain B–M algorithm to find $\underline{e}_{j_k}^{(c)}$.

7) $e = |\underline{e}_0^{(c)}|\underline{e}_1^{(c)}|...|\underline{e}_{n-1}^{(c)}|$ and $c = r \oplus e$.

Example 4.6.1: Consider a 2–D BCH code having the same zero locations as in Example 4.2.1. The received vector r and $R^!$ are as shown below.

$$r = \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 0 \end{array}$$

$$R' = \begin{array}{|ccccc|} \hline 0 & 1 & 1 & 1 & 0 \\ 1 & \alpha^4 & 1 & 1 & \alpha^7 \\ 1 & \alpha^8 & 1 & 1 & \alpha^{14} \\ 1 & \alpha^2 & 1 & 1 & \alpha^{11} \\ 1 & \alpha & 1 & 1 & \alpha^{13} \\ \hline \end{array}$$

Using time–domain B–M algorithm on the first row $(1, \alpha^4, 1, 1, \alpha^7)$ we obtain the auxiliary vector $\lambda = (\alpha^{10}, 0, \alpha^9, \alpha^6, 0)$. As the rows 1 to 4 are conjugates, this auxiliary polynomial is sufficient for decoding purposes. Moreover, it indicates that column 1 and 4 are in error. The extended E^s and e^s are as shown below.

$$E^s = \begin{array}{|ccccc|} \hline 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & \alpha^9 \\ 0 & \alpha^2 & 0 & 0 & \alpha^3 \\ 0 & \alpha^8 & 0 & 0 & \alpha^{12} \\ 0 & \alpha^4 & 0 & 0 & \alpha^6 \\ \hline \end{array}$$

$$e^s = \begin{array}{|ccccc|} \hline 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Using time–domain B–M algorithm on the column–1 we obtain $\lambda = (\alpha^6, 0, \alpha^{10}, 0, \alpha^9)$. The same auxiliary polynomial can be used for the last column. Then the error pattern is found to be

$$e = \begin{array}{|ccccc|} \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

and the code vector is obtained as $r = c \oplus e$.

We now give a comparative study of the hardware complexity, the number of computations and clock times required for decoding a 2–D BCH code using BA–1, BA–2 and and MSTD. It is assumed that a single computation involves a Galois multiplication followed by addition which is completed in one clock time. The DFT's are computed using systolic cell described in [45]. For BA–1 and BA–2, the computation of the common row (column) connection polynomial using B–M algorithm is done serially and the computation of individual rows (columns) are done in parallel. Syndrome extension computations are done concurrently. For MSTD all time–domain B–M algorithm computations are done concurrently. It is possible to have less computations using conjugacy property and fast computation of DFT. However, the magnitudes will have the same order of complexity as given in Table 4.7.1.

Table 4.7.1 Complexity comparisons of various decoding
algorithms for 2–D BCH codes

Algorithm	Hardware	# of Computations	# of Clocks
BA–1	n^2 Cells 1 B-M module n R-E modules	$n^3+n^2t+8t^3$	$2nt+4n+8t^3$
BA–2	$2tn$ Cells t B–M modules 2t R–E modules Chien Computer	$6n^2t+3nt^2+nt+6t^3$	$2nt+5n+4t^3+4t^2+4t$
MSTD	n^2 cells t B–M modules n R-E modules	$9n^2t+8nt^2+nt$	$6n^2+4nt+2n$

BA–1 has smallest delay but requires more hardware and has largest computational burden. Moreover BA–1 has the simplest control circuitry and therefore preferable for small values of t and n. BA–2 has the smallest computations of all, marginally higher delay than BA–1 and moderate hardware complexity. Therefore BA–2 is preferable for large values of t and n. MSTD has least hardware complexity of all, more computations than BA–2 and longest delay. MSTD may be used for moderate values of t and n, if decoding delay requirement not stringent. It may be noted that as MSTD has to do 1–D DFT computation of either all the rows or all the columns, 2–D decoding scheme based on time domain B–M algorithm is not as advantageous as the corresponding 1–D decoding scheme.

4.8 An alternate decoding algorithm for 2–D BCH codes using multiple LFSR synthesis algorithm

We have seen earlier in Section 2.5, Example 2.5.3, that for the class 2–D BCH codes having zero locations in the form of a contiguous 2t x 2t array, the problem of finding error–locator polynomial can also be formulated as a equivalent multiple LFSR synthesis problem. The objective of such a synthesis procedure is to find the common row (column) connection polynomial satisfied by all the rows (columns) of the syndrome window. In this section we employ Feng–Tzeng algorithm for multiple LFSR synthesis to decode the class of 2–D BCH codes and investigate the error–correction capabilities of the decoding algorithm.

The multisequence LFSR synthesis is formulated as follows. Let $\{S_i^{(h)}\}_{i=1}^N$ for $h = 1$ to t, be the t given sequences taking values from F. Let $\sigma(z) = (\sigma_0 + \sigma_1 z + \dots + \sigma_d z^d)$ be the connection polynomial over F, where $\sigma_0 = 1$, but other σ_i's can be zero. Therefore the degree of $\sigma(z)$ can be at most ρ. If

$$S_j^{(h)} + \sigma_1 S_{j-1}^{(h)} + \dots + \sigma_\rho S_{j-\rho}^{(h)} = 0; \quad j = \rho+1, \rho+2, \dots, N \text{ and } h = 1, \tag{4.8.1}$$

then ρ and $\sigma(z)$ completely specify the length and connection polynomial of the LFSR generating all the t given sequences. The objective is to find minimum ρ such that relation (4.8.1) is satisfied. This problem can be rephrased in a slightly different manner by constructing the matrix S_M shown in (4.8.2) from the given multiple sequences. Then the objective is to determine minimum ρ such that $\rho+1$ columns of S_M are linearly dependent and to find the coefficients pertaining to this dependence.

In S_M note that X_i's can be so specified to satisfy the dependency relation (4.8.1). Then the multisequence problem can be viewed as a special case of a more general problem of finding the smallest initial set of dependent columns in a M x N matrix over the field F with rank less than N. Let

$$A = \begin{bmatrix} a_{11} & a_{12} \cdots a_{1N} \\ a_{21} & a_{22} \cdots a_{2N} \\ \vdots & \vdots & \vdots \\ a_{M1} & a_{M2} \cdots a_{MN} \end{bmatrix}$$

be such a matrix. For $0 \leq l < N$, the first $\rho+1$ columns of A are said to be linearly dependent if there exist $c_1, c_2, \dots, c_\rho$ in F, not all zero, such that,

$$a_{i,\rho+1} + c_1 a_{i,\rho} + \dots + c_\rho a_{i,1} = 0, \text{ for } i = 1, 2, \dots, M.$$

In [53], Feng and Tzeng have given an iterative algorithm, which they call as fundamental iterative algorithm, for finding the smallest set of linear dependent columns of a matrix over F. They have also proposed through a refinement of fundamental iterative algorithm, a generalized B–M algorithm for solving the multisequence problem (FT–1). We also like to note that in an earlier paper [52], the same authors have given a generalization of Euclidean algorithm for the same multisequence problem (FT–2). For details of these algorithms refer to [52] and [53]. FT–1 and FT–2 are functionally equivalent though operationally different. For a single sequence LFSR synthesis FT–1 and FT–2 reduce respectively to B–M algorithm and Euclidean algorithm.

$$S_M = \begin{bmatrix}
S_1^{(1)} & S_2^{(1)} & \dots & S_p^{(1)} & S_{p+1}^{(1)} & \dots & S_{N-1}^{(1)} & S_N^{(1)} \\
S_1^{(2)} & S_2^{(2)} & \dots & S_p^{(2)} & S_{p+1}^{(2)} & \dots & S_{N-1}^{(2)} & S_N^{(2)} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
S_1^{(t)} & S_2^{(t)} & \dots & S_p^{(t)} & S_{p+1}^{(t)} & \dots & S_{N-1}^{(t)} & S_N^{(t)} \\
\hline
S_2^{(1)} & S_3^{(1)} & \dots & S_{p+1}^{(1)} & S_{p+2}^{(1)} & \dots & S_N^{(1)} & X_{N+1}^{(1)} \\
S_2^{(2)} & S_2^{(2)} & \dots & S_{p+1}^{(2)} & S_{p+2}^{(2)} & \dots & S_N^{(2)} & X_{N+1}^{(2)} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
S_2^{(t)} & S_3^{(t)} & \dots & S_{p+1}^{(t)} & S_{p+2}^{(t)} & \dots & S_N^{(t)} & X_{N+1}^{(t)} \\
\hline
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
\hline
S_{N-1}^{(1)} & S_N^{(1)} & \dots & X_{N-2+p}^{(1)} & X_{N-1+p}^{(1)} & \dots & X_{2N-3}^{(1)} & X_{2N-2}^{(1)} \\
S_{N-1}^{(2)} & S_N^{(2)} & \dots & X_{N-2+p}^{(2)} & X_{N-1+p}^{(2)} & \dots & X_{2N-3}^{(2)} & X_{2N-2}^{(2)} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
S_{N-1}^{(t)} & S_N^{(t)} & \dots & X_{N-2+p}^{(t)} & X_{N-1+p}^{(t)} & \dots & X_{2N-3}^{(t)} & X_{2N-2}^{(t)} \\
\hline
S_N^{(1)} & X_{N+1}^{(1)} & \dots & X_{N-1+p}^{(1)} & X_{N+p}^{(1)} & \dots & X_{2N-2}^{(1)} & X_{2N-1}^{(1)} \\
S_N^{(2)} & X_{N+1}^{(2)} & \dots & X_{N-1+p}^{(1)} & X_{N+p}^{(1)} & \dots & X_{2N-1}^{(2)} & X_{2N-1}^{(2)} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
S_N^{(t)} & X^{(t)} & \dots & X_{N-1+p}^{(1)} & X_{N+p}^{(1)} & \dots & X_{2N-2}^{(t)} & X_{2N-1}^{(t)}
\end{bmatrix} \qquad (4.8.2)$$

Remark 4.8.1: It is important to understand how the FT–1 (FT–2) work when applied to error–correction. We have from 1–D deconvolution approach that the decoding involves computation of the error locator polynomial from the knowledge of syndromes. This in turn depends on the solution of the linear system of equations formed from syndromes. The cardinality of the smallest possible linearly dependent set is one greater than the rank of the associated syndrome matrix. Then the degree of the connection polynomial is also equal to the rank of the associated syndrome matrix and moreover the connection polynomial is

unique. As multisequence LFSR synthesizing algorithms FT–1 and FT–2 find the smallest initial set of linearly dependent columns, it is clear that the output is the unique connection polynomial having the same degree as the rank of the matrix.

We need the following definition and lemma from [55].

Definition 4.8.1: If $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$ are two vectors define product $a \odot b = (a_1 b_1, ..., a_n b_n)$. If A and B are two m x n matrices the $A \odot B$ is a m^2 x n matrix having as rows all the product of $a \odot b$, where a is row of A and b is a row of B.

Lemma 4.8.1: If a linear combination, with nonzero coefficients, of columns of $A \odot B$ is 0, then

$$rank(A) + rank(B) \leq n$$

We next prove the following important lemma which is useful to determine capabilities of FT–1 (FT–2) to correct different 2–D burst–errors.

Lemma 4.8.2: For a 2–D BCH code with the syndromes $S_{j,j'}$ for $j,j' = 1$ to $2t$ are known. Then FT–1 (FT–2) algorithm followed by recursive extension determines syndromes $S_{j,j'}$ for $j = 1$ to $2t$ and $j' = 1$ to n, for the following cases of 2–D burst–errors.

1) All quasi–cyclic burst–errors upto area t x t.

2) All column–restricted Blahut's 2–D burst–errors upto area t x t.

3) All quasi–cyclic 2–D burst errors $e(x,y) = \sum_{k=1}^{v} e_k(x) y^{jk}$ having area v x v, where $1 \leq v \leq 2t-1$, and $e_j(x) = x^p (\sum_{k=0}^{v-1} e_{j,k} x^k)$ for j=1 to v, considered as a vector space over GF(q) are linearly independent.

Proof. Suppose v columns are in error. Constructing the associated syndrome matrix S_M of dimensions $v(2t-v)$ x $(v+1)$ we have,

$$
S_M = \begin{bmatrix}
\begin{array}{cccc}
S_{1,1} & \cdots & S_{1,v} & S_{1,v+1} \\
S_{1,2} & \cdots & S_{1,v+1} & S_{1,v+2} \\
\cdot & & \cdot & \cdot \\
\cdot & & \cdot & \cdot \\
S_{1,2t-v} & \cdots & S_{1,2t-1} & S_{1,2t}
\end{array} \\
\hline
\begin{array}{cccc}
S_{2,1} & \cdots & S_{2,v} & S_{2,v+1} \\
S_{2,2} & \cdots & S_{2,v+1} & S_{2,v+2} \\
\cdot & & \cdot & \cdot \\
\cdot & & \cdot & \cdot \\
S_{2,2t-v} & \cdots & S_{2,2t-1} & S_{2,2t}
\end{array} \\
\hline
\begin{array}{cccc}
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot
\end{array} \\
\hline
\begin{array}{cccc}
S_{2t,1} & \cdots & S_{2t,v} & S_{2t,v+1} \\
S_{2t,2} & \cdots & S_{2t,v+1} & S_{2t,v+2} \\
\cdot & & \cdot & \cdot \\
\cdot & & \cdot & \cdot \\
S_{2t,2t-v} \cdots & & S_{2t,2t-1} & S_{2t,2t}
\end{array}
\end{bmatrix}
\tag{4.8.3}
$$

Since $e(x,y) = \sum\limits_{k=1}^{v} e_k(x)^{j_k}$, using the notation $e_k(\alpha^i) = X_{i,k}$, $\beta^{j_k} = Y_k$, we factor S_M into

$$
S_M = X_1 Y_1 \tag{4.8.4}
$$

where

$$
Y_1 = \begin{bmatrix}
Y_1 & Y_1^2 & \cdots & Y_1^{v+1} \\
Y_2 & Y_2^2 & \cdots & Y_2^{v+1} \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
Y_v & Y_v^2 & \cdots & Y_v^{v+1}
\end{bmatrix}
\tag{4.8.5}
$$

and

$$X_1 = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,v} \\ Y_1 X_{1,1} & Y_1 X_{1,2} & \cdots & Y_1 X_{1,v} \\ & & & \\ Y_1^{2t-v-1} X_{1,1} & Y_1^{2t-v-1} X_{1,2} & & Y_1^{2t-v-1} X_{1,v} \\ \hline X_{2,1} & X_{2,2} & \cdots & X_{2,v} \\ Y_1 X_{2,1} & Y_1 X_{2,2} & \cdots & Y_1 X_{2,v} \\ & & & \\ Y_1^{2t-v-1} X_{2,1} & Y_1^{2t-v-1} X_{2,2} & & Y_1^{2t-v-1} X_{2,v} \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline X_{2t,1} & X_{2t,2} & \cdots & X_{2t,v} \\ Y_1 X_{2t,1} & Y_1 X_{2t,2} & \cdots & Y_1 X_{2t,v} \\ & & & \\ Y_1^{2t-v-1} X_{2t,1} & Y_1^{2t-v-1} X_{2t,2} & & Y_1^{2t-v-1} X_{2t,v} \end{bmatrix} \qquad (4.8.6)$$

X_1 can be expressed as matrix product $X_2 \odot Y_2$ of two matrices X_2 and Y_2, where X_2 is of dimensions $2t$ x v with $(i,j)^{th}$ element $X_{i,j}$ and Y_2 is of dimensions $(2t-v)$ x v having $(i,j)^{th}$ element Y_j^{i-1}. Therefore (4.8.4) can be rewritten as

$$S_M = (X_2 \odot Y_2) \, Y_1 \qquad (4.8.7)$$

The factorization of X_2 depends on the particular error configuration and therefore we consider several cases.

Case 1: Quasi–cyclic and Blahut's 2–D bursts upto area t x t

Let $v \le t$ columns be in error. For the quasi–cyclic case, $e_j(x) = \sum_{k=1}^{v} e_{j,k} x^{i_k}$ for j=1 to v, and for the Blahut's burst $e_j(x) = x^{p(j)} \sum_{k=1}^{v} e_{j,k} x^{i_k}$ for j=1 to v. Note that Y_2 is a Vandermonde matrix. It is not necessary to know the exact rank of X_2. Therefore

$$\text{rank}(Y_2) = \min(2t-v, \, v) = v; \, \text{rank}(X_2) = v_0 \geq 1$$

Since $\text{rank}(X_2) + \text{rank}(Y_2) > v$, it follows invoking Lemma 4.8.1 that $\text{rank}(X_1) = v$. As X_1 is of full rank, the linear dependence relation among columns of S_M induces the same dependence relation among the columns of Y_1. It is easily seen that the linear dependence among columns of Y_1 gives the connection polynomial whose roots give locators of column errors.

Case 2: Patterned quasi–cyclic bursts of statement 3

Let $v > t$ column be in error. Then

$$\text{rank}(Y_2) = 2t-v; \, \text{rank}(X_2) = v_0;$$

If

$$2t-v+v_0 > v, \text{ or } 2t+v_0 > 2v \tag{4.8.8}$$

then the $\text{rank}(X_1) = v$ and therefore FT–1 (FT–2) produces the correct connection polynomial. We now show that for the case of quasi–cyclic burst in statement 3, v_0 can be shown to be v. Consider a quasi–cyclic 2–D burst with $e_j(x) = x^P(e_{j,0}x^{i_0} + e_{j,1}x^{i_1}+...+e_{k,v-1}x^{i_{v-1}})$ for $j = 1$ to v. Then

$$X_2 = X_3 X_4 e' \tag{4.8.9a}$$

where $X_3 = \text{diag}[X_p, \, X_p^2, \, ..., \, X_p^{2t}]$

$$
X_4 =
\begin{bmatrix}
X_0 & X_1 & ... & X_v \\
X_0^2 & X_3^2 & ... & X_v^2 \\
. & . & ... & . \\
. & . & ... & . \\
. & . & ... & . \\
X_0^{2t} & X_1^{2t} & ... & X_v^{2t}
\end{bmatrix}
\quad
e' =
\begin{bmatrix}
e_{1,0} & e_{2,0} & ... & e_{v,0} \\
e_{1,1} & e_{2,1} & ... & e_{v,1} \\
. & . & ... & . \\
. & . & ... & . \\
. & . & ... & . \\
e_{1,v-1} & e_{2,v-1} & ... & e_{v,v-1}
\end{bmatrix}
\tag{4.8.9b}
$$

where $X_p = \alpha^P$ and $X_k = \alpha^{jk}$ for $k = 0$ to v–1. As X_3 is a diagonal matrix, $\text{rank}(X_3 X_4) = \text{rank}(X_4)$. Since X_4 is a 2t x v Vandermonde matrix, $\text{rank}(X_4) = \min(2t,v) = v$. By hypothesis e' is v x v invertible matrix. Hence $\text{rank}(X_2) = v$, as required and the relation (4.8.8) is satisfied. Therefore FT–1 (FT–2) produces the correct connection polynomial. $\square$

The following examples show that, in addition to bursts of Lemma 4.8.2, there are other patterns of quasi–cyclic bursts for which FT algorithms produce the correct connection polynomial.

Example 4.8.1: Consider 2–D BCH code of area 5 x 5, with $C_{j,j'} = 0$, for $1 \leq j \leq 4$, $1 \leq j' \leq 4$. For two error patterns $e_1(x,y) = (xy + x^2y^2 + x^3y^3)$ and $e_2(x,y) = (x^2y + (x+x^2)(y^2+y^3))$ the syndromes are shown below.

$$
S_1 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & \\
\hline
1 & a^7 & a^{13} & a^{14} & 1 & - \\
2 & a^{13} & a^{14} & 1 & a^{11} & - \\
3 & a^{14} & 1 & a^{11} & a^7 & - \\
4 & 1 & a^{11} & a^7 & a^{13} & - \\
 & - & - & - & - & - \\
\end{array}
\qquad
S_2 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & \\
\hline
1 & 1 & 0 & a^{11} & a^4 & - \\
2 & a^7 & 1 & a^8 & 0 & - \\
3 & 0 & a^2 & 1 & a^{13} & - \\
4 & a & a^{14} & 0 & 1 & - \\
 & - & - & - & - & - \\
\end{array}
$$

Suppose decoding starts with computation of the row connection polynomial. Constructing the matrices e_1' and e_2' as in Equation (4.8.9b), we have

$$
e_1' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\qquad\qquad
e_2' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
$$

The rank$(X_2) = 3$ for e_1' and rank$(X_2) = 2$ for e_2'. From Equation (4.8.8) it is seen that for syndrome array S_1 the unique connection polynomial is found to be $(z^3 + a^{11}z^2 + a^2z + a^3)$. In the case of syndrome array S_2 non–unique connection polynomial is produced.

Example 4.8.2: Consider a 2–D BCH code of area 7 x 7 with the zeros $C_{j,j'} = 0$, for $1 \leq j \leq 6$ and $1 \leq j' \leq 6$. The error pattern is $e_3(x,y) = y^5 + x^2(1+y^2+y^3) + x^3(1+y^2)$ which is a 3 x 4 quasi–cyclic error burst and the syndrome array is

$$
S_3 = \begin{array}{c|ccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & \\
\hline
1 & a^2 & a^5 & a^3 & a & 1 & a & - \\
2 & a^2 & a^4 & 1 & a^3 & a^2 & a^6 & - \\
3 & a^3 & a^6 & 0 & 1 & a & a & - \\
4 & a^6 & a^4 & a^4 & a & a^5 & 1 & - \\
5 & 1 & a^6 & a^2 & a^5 & a^2 & 0 & - \\
6 & a^3 & 1 & a^4 & a^5 & 0 & a^4 & - \\
 & - & - & - & - & - & - & - \\
\end{array}
$$

The error matrix $e_3^!$ is

$$e_3^! = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Since the rank(X_2) = 3 for $e_3^!$, the relationship (4.8.8) is satisfied and the FT algorithms produce the unique connection polynomial ($z^4 + z^3 + \alpha^2 z^2 + \alpha^5 z + \alpha^3$).

Theorem 4.8.1: Consider a 2–D BCH code with the zero locations $C_{j,j'} = 0$, for $j,j' = 1$ to 2t. Then FT–1 (FT–2) algorithm based 2–D BCH decoding can correct the following cases of 2–D burst–errors.

1) All quasi–cyclic burst–errors upto area t x t.

2) All column–restricted Blahut's 2–D burst–errors upto area t x t.

3) All quasi–cyclic 2–D burst errors $e(x,y) = \sum\limits_{k=1}^{v} e_k(x)y^{jk}$ having area v x v, where $1 \leq v \leq 2t-1$, and

   $e_j(x) = x^p(\sum\limits_{k=0}^{v-1} e_{j,k}x^k)$ for j=1 to v, considered as a vector space over GF(q) are linearly independent.

Proof. From Lemma 4.8.2 it follows that $S_{j,j'}$ for j = 1 to 2t and j' = 1 to n are determined from recursive extension using the common row connection polynomial. There are several options to do the column processing depending upon the type of burst–error correction. One way is to again find the common column connection polynomial using FT algorithms on the 2t column sequences of the syndrome window. This method is applicable to correcting quasi–cyclic burst–errors upto area v x t and some patterned bursts upto area 2t–1 x 2t–1 indicated in statement 3 of Theorem 4.8.1. For the latter patterned burst–error case, the FT algorithms produces the unique column connection polynomial, as the error matrix for the column sequences is transpose of the corresponding error matrix for the row sequences and hence has the same rank. The remaining syndromes are found using the common column connection polynomial by recursion. Another way to complete the decoding is to find the columns which are in error and process them individually. The latter method is appropriate when Blahut's 2–D bursts are to be corrected. □

Multiple Sequence LFSR Based Decoding Algorithm for 2–D BCH Codes

1) Pass row syndrome sequences $\{S_{i,j}\}_{j=1}^{2t}$ for i=1 to 2t, through FT–1 (FT–2) and find the row connection polynomial $\Lambda_r(z)$. Compute $\{S_{i,j}\}_{j=1}^{n}$ for i = 1 to 2t, by recursive extension.

2) For correcting quasi–cyclic bursts, pass column syndromes $\{S_{i,j}\}_{i=1}^{2t}$ for j=1 to 2t, through FT–1 (FT–2) algorithm and find the column connection polynomial $\Lambda_c(z)$. Compute $\{S_{i,j}\}_{i=1}^{n}$ for j= 1 to n, by syndrome extension using $\Lambda_c(z)$.

For correcting Blahut's bursts find the roots of the row connection polynomial $\Lambda_r(z)$ and identify the columns in error. Compute 1–D IDFT of each of the 2t rows of the syndrome window at every column error locations. This determines $\{E'_{j,p_i}\}_{j=1}^{2t}$ for i = 1 to v, where p_i denotes the position of ith column error and v ≤ t. Process them individually to obtain $\{E'_{j,p_i}\}_{j=0}^{n-1}$ for i = 1 to v.

3) For correction of quasi–cyclic burst IDFT S gives e.
For Blahut's burst case, 1–D IDFTs of $\underline{E}'^{(c)}_{p_i}$ for i = 1 to v, gives e.

The following theorem gives error–correction ability of FT algorithm for the class of Γ 2–D BCH codes.

Theorem 4.8.2: For a Γ 2–D BCH code the syndromes $S_{j,j'}$ for j = 1 to t and j' = 1 to 2t are known. Then the FT algorithm followed by recursive extension can determine syndromes $S_{j,j'}$ for j = 1 to n and j' = 1 to n, for the following cases of 2–D burst–errors.

1) All quasi–cyclic burst–errors upto area t x t.
2) All column restricted Blahut's 2–D bursts upto area t x t.

Proof : Proofs are the same as in Case–1 of Theorem 4.8.1. The burst–errors considered in Case–2 of Theorem 4.8.1 are not correctable because the rank(X_4) (see Equation (4.8.9b)) is upper bounded by 't' instead of '2t' and hence the relationship (4.8.8) is not satisfied. □

Remark 4.8.2: It is seen that FT algorithm based decoding for 2–D BCH code can correct all types of burst–error correctable using BA–1 and BA–2, and a few more types of burst–errors not correctable by the latter. Therefore, it is best to decide the choice of the FT based or BA based 2–D BCH decoding depending upon the predominant channel error characteristics. For Γ 2–D code both IBA and FT algorithm based decoding have the same performance. Therefore the choice is made based on the implementation complexity considerations.

4.9 A class of random error–correcting 2–D BCH codes

We have so far studied some interesting classes of 2–D BCH codes that are naturally suited for burst–error correction. Though 2–D BCH codes have poor random error–correction capability, it is still instructive to study the requirements on the zero locations with regard to just random error correction. In what follows we introduce a class of t–random error–correcting (t–REC) 2–D BCH code and give a decoding algorithm to correct upto designed error–correcting capability. We also make a comparative study of requirements on the zero locations for 2–D BCH codes with regard to correction of different sorts of random and burst–errors.

We start with the following lemma giving the basic configuration of zero location for t–REC 2–D code.

Lemma 4.9.1: A 2–D BCH code having zero locations

$$C_{1,j} = 0, \text{ for } j = 1,...,2t$$

$$C_{i,1} = 0, \text{ for } i = 1,...,2t$$

$$C_{i,j} = 0, \text{ for } i = 2,...,t \text{ and } j = 1,...,t-i+2$$

has unique syndrome pattern for all the random error patterns of weight upto t.

Proof. Suppose that two different random errors b_1 and b_2 have the same syndrome. Then $b' = b_1 - b_2$ is a codeword and let B_1, B_2 and B' represent 2–D DFTs of b_1, b_2 and b' respectively. If the number of nonzero columns of b' exceed t, then at least one nonzero column of b' has weight 1. Since the degree of row connection polynomial for B' is at most 2t, by complexity theorem, B' has first row all zeros, and this implies that a nonzero column of b' must have weight at least 2. Then the number of nonzero columns of b' is $\leq$ t and therefore the degree of the row connection polynomial of B' is at most t. Again by 2–D complexity theorem two rows of B' are zeros and this implies that a nonzero column of b' must have weight at least 3. Therefore b' has at most $\lfloor 2t/3 \rfloor$ nonzero columns. For $t \leq 3$ there can be at most two nonzero columns of b' having weight > t. Clearly this cannot be the case. For $t > 3$, $\lfloor 2t/3 \rfloor < t-1$ is satisfied and this further implies that a nonzero column of b' must have weight at least 4. Continuing this way, for r^{th} row, we arrive at the conclusion that a nonzero column of b' must have weight at least r and therefore b' has at most $\lfloor 2t/r \rfloor$ columns. Now $t \leq r$ is seen to be not the case as before. For $t > r$ we have the following inequality

$$\lfloor 2t/r \rfloor \leq t-r+2$$

But then this implies a nonzero column of b' to have weight > r. In any case, we always obtain a contradiction if we assume that b' is a codeword. This completes the proof. □

Theorem 4.9.1: For a t–REC 2–D BCH code of Definition (4.9.1), B–M algorithm processing sequentially the syndromes of the rows i = 1 to t, followed by syndrome extension determines the syndromes $\{S_{i,j}\}_{j=1}^{n}$ for i = 1 to t, if v ≤ t errors happen. Similar processing of columns determines entire syndrome array.

Proof: If the B–M algorithm on the $\{S_{1,j}\}_{j=1}^{2t}$ produces the connection polynomial of degree v ≤ t, then we are through. If there is a column error which has zero contribution for the first row, then the degree of connection polynomial for the first row is upper bounded by t–2. The first row connection polynomial can be treated as virtual erasure polynomial when B–M algorithm processes the syndromes of the second row. It is known that t consecutive zero positions are required for the B–M algorithm to correct t–2 erasures and a single error. As second row has t consecutive zeros, we see from Lemma 4.2.1 that B–M algorithm initialized with first row connection polynomial produces the correct connection polynomial. Similarly, if there are p columns having zero contribution to the first row and each is of weight 2, then there are at most t–2p erasures and p errors and t consecutive zero location are enough to find the connection polynomial using B–M algorithm. In general, for k ≥ 1 and m = 2 to t, when processing row m, for k column errors having weight m which have zero contributions to first m–1 rows we have at most (t–km) erasures and k errors which require

$$t- km +2k = t - k(m-2) \leq t ,$$

zero locations to find the connection polynomial. Hence B–M algorithm processing sequentially all the t rows of the syndrome window produces the required connection polynomial. The remaining syndromes of t rows are computed from recursive extension.

Now the syndromes $S_{1,j}$ for j = 1 to 2t and $S_{i,j}$ for i = 2 to n and j = 1 to t, are known. Applying B–M algorithm to t consecutive column sequences of the syndromes window the column connection polynomial is computed. Then rest of the syndromes are computed from syndrome extension. □

Decoding Algorithm for t–REC Codes

1) Pass $\{S_{1,j}\}_{j=1}^{2t}$ and $\{S_{r,j}\}_{j=1}^{t}$ for r = 2 to t, sequentially through B–M algorithm and find $\Lambda_r(z)$.

2) Use $\Lambda_r(z)$ to find $\{S_{r,j}\}_{j=1}^{t}$ for r = 1 to n by recursive extension.

3) Pass $\{S_{j,1}\}_{j=1}^{2t}$ and $\{S_{j,r}\}_{j=1}^{t}$ for r = 2 to t, sequentially through B–M algorithm and find $\Lambda_c(z)$.

4) Use $\Lambda_c(z)$ to find $\{S_{r,j}\}_{j=1}^{n}$ for r = 1 to n, by recursive extension

5) Compute 2–D IDFT of $\{R_{i,j}-S_{i,j}\}$ to determine the codeword c.

Example 4.9.1: Consider a 5 x 5, 2–D BCH code over $GF(2^4)$. For the error pattern $e(x,y) = \alpha xy + x^2 y^2$, the complete syndrome array is as shown in Figure 4.9.2. The locations within the bordered area represent

available syndromes corresponding to zero locations.

$$
S = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 0 \\
1 & \alpha^2 & \alpha^{12} & \alpha^{10} & \alpha^4 & \alpha^{12} \\
2 & \alpha^{12} & \alpha^{10} & \alpha^4 & \alpha^{12} & \alpha^2 \\
3 & \alpha^{10} & \alpha^4 & \alpha^{12} & \alpha^2 & \alpha^{12} \\
4 & \alpha^4 & \alpha^{12} & \alpha^2 & \alpha^{12} & \alpha^{10} \\
0 & \alpha^{12} & \alpha^2 & \alpha^{12} & \alpha^{10} & \alpha^4
\end{array}
$$

Figure 4.9.2 Completed syndrome array

By passing $S_{1,1}$ to $S_{1,4}$ through B–M algorithm we find $\Lambda_r(z) = (1 + \alpha^2 z + \alpha^9 z^2)$. Since this is of degree 2 no more passes though B–M algorithm are necessary. Using $\Lambda_r(z)$ find the syndromes of row–2. By passing $S_{1,1}$ to $S_{4,1}$ through B–M algorithm we find $\Lambda_c(z) = (1 + \alpha^2 z + \alpha^9 z^2)$. The remaining syndromes of all the columns are found by using $\Lambda_c(z)$.

Remark 4.9.2: It is clear that in order to correct t random error it is required to have t x t block of zero in addition to some more. But in order to correct t x t 2–D cyclic burst–error the minimum number of zero locations required is $3t^2$, consisting of three t^2 blocks, which corresponds to Γ 2–D BCH code. This has t^2 excess zero locations compared to optimum $2t^2$ required for t x t 2–D cyclic burst–error correction. Increasing further the number of zeros to $4t^2$ consisting of four t^2 blocks does not alter the error–correction properties. It only helps in making the decoding operation more efficient by cutting down some of the computations.

Chapter 5

SPECTRAL DOMAIN DECODING ALGORITHMS
FOR 1-D CYCLIC CODES

5.1 Review of decoding algorithms for non–BCH cyclic codes

The class of BCH codes has proved to be very useful at moderate code lengths. Furthermore, these codes are endowed with a structure that renders them to efficient and simple decoding by using B–M algorithm for LFSR synthesis. However, there are other non–BCH cyclic codes which are not as structured as BCH codes, nevertheless have comparable performances. Therefore, it is of practical interest to find good decoding algorithms for such non–BCH cyclic codes. Some attempts to develop decoding algorithm for such codes have been made earlier in the literature [10], [73].

Blahut has discussed modifications of standard BCH decoding algorithm to decode beyond the designed distance [2]–[3]. The basic idea is to introduce as unknowns extra syndromes (or extra discrepancies) and let B–M algorithm continue to compute connection polynomial as a function of these unknowns. The unknowns are then selected in a systematic way by trial and error, to obtain the smallest weight error vector in the symbol field. The demerit of the scheme is that the complexity increases rapidly with the number of extra errors to be corrected.

Bours et.al [10] have employed the Newton identities in an interesting way to give decoding algorithms for cyclic codes. Their approach is algebraic in nature involving setting up of several equations, eliminating some terms and testing dependency among the equations for different error patterns to arrive at the decoding algorithm. The disadvantage is that algebra involved may be tedious.

5.2 General outlines of 2-D spectral decoding algorithms
for 1-D cyclic codes

An entirely different approach to decoding is possible based on 2–D DFT characterization of a 1–D cyclic code. To understand and design the decoding algorithms 1–D deconvolution approach presented in Chapter 2 is very helpful. The main idea is to analyze the changes in the degree of the connection polynomials due to different configurations of an error pattern of a given weight, upto some limit which should desirably coincide with the correcting limit $\lfloor (d_{min} - 1)/2 \rfloor$ of the code. So the decoding algorithm proceeds in stages anticipating various possible error configurations, and finding appropriate connection polynomials, and testing for their validity. The validity tests are integral part of 2–D decoding algorithms as they have to reject a hypothesized error configuration before proceeding to test for another possible error configuration. The validity tests for the 2–D decoding algorithm are the same as those employed in a

BCH decoder that uses B–M algorithm for LFSR synthesis followed by recursive syndrome extension. The objective of such tests is to check whether the number of distinct roots of the connection polynomial is equal to its degree and whether the syndromes generated from recursive extension satisfy conjugacy constraints. The first test is shown to be equivalent to periodicity test on the generated syndrome sequence (Theorem 9.6.1, [4]). Therefore for the case of 2–D decoding algorithms validating tests on a connection polynomial include periodicity tests on the generated row (column) syndrome sequence, testing for conjugacy constraints and testing for consistency of the generated row syndrome with the already known syndromes along that row (column).

A 2–D spectral algorithm has more flexibility compared to 1–D spectral algorithm in that one can use two possible connection polynomials and two possible directions for recursive extension. This also makes a 2–D decoding algorithm handle codes which are not easily decodable using 1–D techniques. The main effort is therefore how to find the appropriate row or column connection polynomial for a given configuration of error using all possible information from the syndromes. This emphasis is motivated from the engineering viewpoint and helps in a better understanding of the algorithm. Therefore, this approach may be seen as combining to some extent the advantages of both Blahut's method and algebraic method due to Bours et.al. The price paid for this advantage over 1–D case is that the decoding algorithm becomes complex.

We now consider some important features of the 2–D spectral decoding algorithm for non–BCH cyclic codes. A novel feature is interesting combination of Chien search method and systematic trial–and–error method. For example, there arise situations where an error may have affected 3 columns and it is possible to determine only two affected columns by algebraic methods. A new method possible in the case of 2–D decoding algorithms is to construct different connection polynomials of degree 3, by multiplying each of the column locators not known to be corrupted and known partial error locator polynomial and test for validity of the constructed connection polynomial. 2–D complexity theorem ensures that only one of the constructed polynomials passes the validity tests if less than $\lfloor (d_{min}-1)/2 \rfloor$ errors have occurred.

Sometimes Blahut's method of guessing unknown syndromes comes in handy. As each guess requires tests to ascertain validity the number of possible guesses should be reasonable. The justification again rests on 2–D complexity theorem which ensures unique choice of estimated syndrome if $\lfloor (d_{min}-1)/2 \rfloor$ errors have occurred. Another advantage of 2–D spectral techniques is that computationally expensive trial–and–error techniques are sparingly used only for some cases of error configurations.

Another important feature of 2–D spectral algorithms, which we do not consider in detail, is erasure correction possibility. This will be made clear from some examples for which errors–and–erasures

decoding is discussed. These 2–D spectral decoding algorithms particularly are best suited to correction of periodic erasure bursts.

5.2.1 Classification of 2-D decoding algorithms

For ease of understanding it is useful to classify 2–D spectral decoding algorithms into 3 broad categories depending upon implementation complexity and computational burden considerations as: type–1, type–2 and type–3.

Type–1 2–D spectral decoding algorithm is one in which finding connection polynomials and doing recursive extension are straightforward. The important steps of type–1 decoding algorithm include one or more passes through B–M algorithm to find row connection polynomials followed by row recursion and similar routines in the column direction. This type of 2–D decoding algorithm, in general, has little more implementation complexity than a BCH decoder.

A 2–D decoding algorithm of type–2 decoding algorithm requires estimation of either syndromes or connection polynomial (in the sense explained in Section 5.2). For such a decoding algorithm it is important to confirm by 2–D complexity theorem arguments that only one choice works. As each guess involves validating tests both the computational burden and implementation complexity are greater than type–1 but less than type–3.

A 2–D decoding algorithm is called type–3 if the decoding algorithm cannot correct upto $\lfloor (d_{min} -1)/2 \rfloor$ at one shot. A characteristic feature of such 2–D cyclic codes is that their spectrum has several complete rows (columns) of zero locations. It can be seen from [56] that 1–D and 2–D cyclic codes discussed in this thesis can be interpreted as concatenated codes. Therefore for the case of 2–D codes having type–3 decoding algorithm the row of zero locations give a clue to the component codes of the concatenation scheme. It also follows that row (column) vectors of such 2–D cyclic codes belong to subcodes which may have some error–correction capability.

There is a lot of literature concerning decoding algorithms for concatenated codes because of the practical importance of concatenated codes for correction of complicated type of errors [83]–[84]. A typical concatenated code consists of outer Reed–Solomon codes over extension field whose symbols are are information bits for the inner code. The encoding scheme consists of two stages of outer code encoding followed by inner code encoding. The decoding scheme for concatenated codes bears a resemblance to decoding scheme for product codes [4]. It is interesting to note that the lower bound on d_{min} for a concatenated code as the product of d_{min} of its component codes is similar to the lower bound on d_{min} for a product code as the product of d_{min} of its component codes, and this fact is crucial to understanding the similarities in decoding algorithms. The decoding of a concatenated algorithm consists of two stages: In the

first stage inner codes are decoded on the basis of a maximum likelihood decoder; The second stage consists of errors–and–erasures decoding of outer code on the basis of side information available based on the first stage inner code decoding. It is important to note that latter part requires error–and–erasure decoder.

A 2–D decoding algorithm of type–3 differs only in minor respects from a typical concatenated decoding algorithm. The condition of requirement of error–and–erasure decoding algorithm is met as the main tool of 2–D decoding algorithms is B–M algorithm for LFSR synthesis which is naturally suited to such purposes. Only difference in decoding may result from the specific details of encoding which is taken into account in some of concatenated algorithms [83]–[84]. 2–D spectral method is not tied to any specific encoding and this may be used to advantage.

5.2.2 Notation for 2–D decoding algorithms

The description of decoding algorithm is preceded by relevant details of zero locations of the code. The figure of configuration of zero locations (The zero locations are marked as 'Z') helps in visualizing how decoding algorithm is progressing. Unless otherwise specified, decoding algorithms are given only for error–correction.

For 2–D decoding algorithms of type–1 and type–2, R, the 2–D DFT of the received vector r, is the input to the algorithm and error vector e is the output. For a type–3 decoding algorithm 1–D DFT of either rows or columns of the r, denoted by R', is the input and the e is the output.

The details of decoding steps follows with explanation of what error configurations motivated it. If some of the steps have used new decoding techniques then formal proofs are given to justify the steps. It is assumed that at each step validity tests are carried out before proceeding to next decoding step even when not stated explicitly. Validity tests consists of periodicity tests on generated syndrome subsequences and checking for consistency of known syndromes with the generated syndromes. If none of the decoding steps produce valid connection polynomial then this is a symptom of occurrence of more number of errors than can be corrected by the code.

The polynomial notation represents rows by powers of 'x' and the columns by powers of 'y'. $'y^{j_k}e_k(x)'$ denotes a column error pattern $e_k(x)$ at j_k^{th} column. The contribution of j_k^{th} column error to syndromes in i-th row is nonzero if $e_k(\alpha^i) \neq 0$. This is important to understand how this term modifies syndromes in different rows.

It is useful to have a representation scheme for various error configurations that help in understanding how B–M algorithm operating on row (column) syndrome sequences takes into account column (row) errors. A column error pattern is represented by the number of errors that occurred in that column. The actual location of errors within a column is not important. For example, if a column error

pattern affects only one row then this column error configuration is represented by (o); if the column error affects two rows the configuration is represented by $\binom{o}{o}$; similarly the column error affecting three rows is represented by $\binom{o}{\substack{o\\o}}$ and so on (One can use numbers instead of stacking 'o'). A column–wise representation of a 2–D error pattern consists of configuration representations of nonzero column error patterns separated by commas. For example, suppose an error has 2 nonzero columns and further each nonzero column contains a single error. Then the configuration is represented by (o,o). A 2–D error can be row–wise represented similarly in terms of row error patterns. Usually a 2–D error pattern is represented using column–wise configuration without explicit mention. The row–wise error representation is explicitly mentioned whenever used. The need for row–wise representation is required for explaining type–3 decoding algorithms.

$\Lambda(z)$ represents the common row connection polynomial and $\deg(\Lambda(z))$ denotes degree of the connection polynomial. The format of $\Lambda(z)$ produced for a given syndrome sequences is the same as in the version of B–M algorithm given in [4], unless otherwise stated. We have seen that computation of final $\Lambda(z)$ involves passing several row–syndrome sequences through B–M algorithm with the proper initialization. In order to avoid too many notations required to represent intermediate connection polynomials, the following convention is followed. If $\Lambda_{m-1}(z)$ generating previous (m–1) row syndrome sequences is known, and given m^{th} row syndrome sequence, then <u>updating of $\Lambda(z)$</u> for m^{th} syndrome sequence means finding $\Lambda_m(z)$ from m^{th} row syndrome sequence by initializing the B–M algorithm with $\Lambda_{m-1}(z)$. The common column connection polynomial is denoted by $\phi(z)$ and <u>updating of $\phi(z)$</u> is explained in terms of $\phi_m(z)$ in a similar manner to updating of $\Lambda(z)$.

The recursive extension is performed in the row direction by $\Lambda(z)$ and in the column direction by $\phi(z)$. The details for regarding recursive extension are not given when there are enough consecutive syndromes available. If there are some unknown syndromes (or breaks) in between, then relevant details to complete recursive extension are given. Sometimes, when all the syndromes in a row are completely known, one can invoke conjugacy property to determine syndromes in the other conjugate rows. Alternatively, conjugacy property can be used to find unknown syndromes so that enough consecutive syndromes are available for performing row or column recursive extension.

In a type–3 decoding algorithm the result of first stage decoding on j^{th} row (column) is assigned a correction number denoted by 'w_j' which is measure of reliability of row–decoding. Lower w_j means decoder output is more reliable. Occasionally, the notation $\Lambda_{row}(z)$ is used to denote connection polynomial of individual row or column syndromes in the first stage of type–3 decoding. The second stage of decoding processes 1–D DFT of columns (or rows) of R' which is denoted by S. The contribution to

components of S at zero locations is entirely due to error. These known syndromes are then used to find appropriate connection polynomials. The components of S at nonzero locations are replaced by syndromes generated by recursive extension of known syndromes. In the second stage decoding, the computation of $\phi(z)$ depends on identifying least reliable columns. Sometimes $\phi_{uc}(z)$ is used to denote column connection polynomial whose roots are locators of rows which are detected as uncorrectable. For some decoding algorithms which make use of row (column) parity-checks, p denotes row (column) parity-check vector.

2-D decoding algorithms are given for several cyclic codes of length 21, 33, 35, 39, 45, 51 and 63 as listed in Table 5.2.1. The codes are numbered according to table in [55]. We do not discuss separate decoding algorithms for even weight subcodes unless some simplifications can be found. Table also lists e_{BCH}, e_{alg} and e_{2-D} denoting respectively error-correction capability of standard BCH decoding algorithm, algebraic decoding algorithm of [10] and 2-D spectral decoding algorithm.

5.3 Decoding algorithms for codes of length 21

5.3.1 Decoding of $(21, 7, 8)$

This code has $e_{BCH} = 2$ and $e_{alg} = 3$. We now show that e_{2-D} is also 3, and decoding is of type-1. The generator polynomial is given by $g(x) = m_1(x)m_3(x)m_7(x)m_9(x)$. The zero locations of the code are as shown in Figure 5.3.1. Z represents zero locations of the code.

r\c	0	1	2	3	4	5	6
0		Z	Z	Z	Z	Z	Z
1	Z	Z	Z		Z		
2	Z	Z	Z		Z		

Figure 5.3.1 Configuration of zeros of $(21, 7, 8)$

Derivation of decoding algorithm: A particular column error pattern $y^{j_k}e_k(x)$ contributes to the ith row syndromes if $e_k(\alpha^i) \neq 0$. The $e_k(x)$ can be of Hamming weight either 1 or 2 or 3. If the weight of $e_k(x)$ is 1, then j_k-th column contributes to all the rows; If the weight is 2, then its contribution is nonzero for rows 1 and 2 and zero for row 0; If it is 3, then its contribution for row 0 is nonzero, otherwise it is zero.

In the step 1, the error configurations (o) or $\binom{o}{o},o)$ or (o,o) or (o,o,o) or $\binom{o}{o},o)$, produce nontrivial $\Lambda(z)$ from 6 consecutive syndromes of row-0. For the last three configuration of errors the remaining syndromes can be computed using $\Lambda(z)$ by recursive extension.

Table 5.2.1 1–D Cyclic Codes having 2–D Spectral Decoding Algorithms

No. in [55]	Parameters of the code			Leaders of zero conjugacy class	Error–Correction capabilities			2–D Algm. Type
	n	k	d		e_{BCH}	e_{alg}	e_{2-D}	
9	21	7	8	1,3,7,9	2	3	3	1
8	21	9	8	0,1,3,7	2	2	3	2
25	33	13	10	1,3	2	4	4	1
26	33	11	11	1,3,11	3	4	5	2
32	35	20	6	1,5	2	2	2	1
36	35	16	7	1,5,7	2	3	3	1
37	35	15	8	0,1,5,7	2	3	3	1
40	35	7	14	0,1,3,5	5	5	6	3
45	39	15	10	1,3	3	4	4	1
47	39	13	12	1,3,13	3	5	5	2
75	45	21	8	1,5,9,15	3	3	3	1
85	45	15	10	1,7,9,15	3	3	4	2
86	45	14	10	0,1,7,9,15	3	3	4	1
88	45	12	10	0,1,3,7,9	3	3	4	1
89	45	11	9	1,3,7,15,21	3	3	4	1
90	45	9	12	1,5,7,9,15	4	4	5	1
91	45	8	12	0,1,5,7,9,15	4	4	5	1
96	51	35	5	1,9	1	1	2	1
97	51	34	6	0,1,9	1	1	2	1
98	51	34	6	0,1,5	1	2	2	1
106	51	27	8	1,3,9	2	2	3	2
111	51	25	8	1,3,9,17	2	2	3	1
117	51	19	10	1,3,9,19	2	2	4	3
121	51	17	12	1,3,9,17,19	2	2	5	1
128	51	11	15	1,3,5,11,19	4	4	7	3
*1	63	24	16	0,1,5,7,9,15,23,21,27	2	2	7	3
*2	63	39	8	1,3,15,31	2	2	3	1

*1 Taken from [56]; *2 Taken from [53]

If the $\deg(\Lambda(z)) = 0$ or 1, then possible configuration of errors can be (o) or $\binom{\text{o}}{\text{o}}$ or $\binom{\text{o}}{\text{o}},\text{o})$. For the second case 3 consecutive syndromes of row–1 easily produce $\Lambda(z)$. For the last case we have the partial knowledge of the location of single error. Then connection polynomial $\Lambda(z)$ is the virtual erasure polynomial for the syndromes of rows 1 and 2. From Lemma 4.2.1 using $\Lambda(z)$ for initialization of B–M algorithm, the sequence of 3 consecutive syndromes in row–1 (row–2) are sufficient to give the correct connection polynomial (3 syndromes are required to correct one error and one erasure). This explains the step 2 of the decoding algorithm.

Decoding Algorithm for (21, 7, 8)

1) Pass $S_{0,1}$ to $S_{0,6}$ through B–M algorithm and find the $\Lambda(z)$. Compute $S_{0,0}$. Go to step 3.

2) If the $\deg(\Lambda(z)) \leq 1$, update $\Lambda(z)$ by passing $S_{1,0}$ to $S_{1,2}$ through B–M algorithm.

3) Determine the the remaining syndromes by recursive extension using $\Lambda(z)$.

4) 2–D IDFT of the S gives e.

Remark 5.3.1: Decoding algorithm is easy to visualize and understand geometrically. The algebra involved is modest in comparison to the algorithm given in [10]. Further, this algorithm can also correct some errors of weight 4 having error configuration $(\text{o},\substack{\text{o}\\\text{o}})$, which is not possible by the algorithm given in [10]. Moreover the proposed algorithm has scope for parallel implementation.

Remark 5.3.2: A closer examination of zero location structure reveals that codes of area 3 x n having 2t continuous zero locations in row–0 and t zero locations in row–1 or row–2 have at least t–random error–correcting capability. The lower bound on minimum distance is derived as follows. If there is a codeword of weight $\leq 2t$, then 2t consecutive syndromes of row–0 by 2–D complexity theorem ensures that non–zero columns of the codeword should have weight at least 2. But in such a case t consecutive syndromes by 2–D complexity theorem ensures that codeword has to be zero. The decoding scheme is easy to generalize. First pass 2t row–0 syndromes through B–M algorithm and find $\Lambda(z)$. If $\deg(\Lambda(z)) \leq t-2$ then update $\Lambda(z)$ by passing t consecutive syndromes through B–M algorithm. It is easy to see that if there are $x \leq t$ columns of even weight then $\deg(\Lambda(z)) = t-2x$ in the first pass. We require at most $(t-2x)+2x = t$ consecutive zero locations in the second pass to take into account x even weight columns.

Example 5.3.1: Consider that a codeword $1\,1\,0\,1\,0\,0\,1$ / $1\,1\,1\,0\,1\,0\,0$ / $0\,0\,1\,1\,1\,0\,1$ is transmitted. The DFT domain is $GF(2^6)$ generated by α a primitive 63rd root of unity with the minimum polynomial (x^6+x+1).

a) For the $e(x,y) = (y^5 + xy + x^2y)$, the available syndromes are indicated by bold and italicized entries.

r\c	0	1	2	3	4	5	6
0	1	α^{45}	α^{27}	α^9	α^{54}	α^{36}	α^{18}
1	0	1	1	α^{36}	1	α^{18}	α^9
2	0	1	1	α^{36}	1	α^{18}	α^9

Using B–M algorithm on row–0 syndromes $\Lambda(z) = (1 + \alpha^{45}z)$. $S_{0,0}$ is found to be 1 using recursive extension. B–M algorithm on the sequence $(0, 1, 1)$ with the initialization $(1 + z\alpha^{45})$ updates $\Lambda(z)$ to $(1 + z + \alpha^{54}z^2)$. The syndromes found from recursive extension are shown in non–bold and non–italicized entries.

b) For $e(x,y) = (y^5 + x + x^2y)$, the syndrome array is shown below

r\c	0	1	2	3	4	5	6
0	1	0	0	α^{54}	0	α^{27}	α^{45}
1	0	α^{52}	α^8	α^5	α^2	α^{20}	α^{17}
2	0	α^4	α	α^{40}	α^{16}	α^{34}	α^{10}

B–M algorithm on row–0 sequence computes $\Lambda(z) = (1 + \alpha^{36}z^2 + \alpha^{54}z^3)$. Recursive extension is straightforward.

c) $e(x,y) = (1 + x + x^2)$ and the syndromes are as shown

r\c	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0.	0

$\Lambda(z)$ is found to be $(1 + z)$, and recursion is straightforward.

For all the examples $C = R \oplus S$, and 2–D IDFT of C gives c.

Erasure decoding of $(21, 7, 8)$

In order to correct both erasures–and–erasures a different approach is needed. The decoding algorithm turns out to be a version of type–3. The first stage decoding is based on $(7, 4, 3)$ code. A row is correctly decoded if it has at most 2 erasures or a single error. Other row–error patterns either go undetected or detected as uncorrectable or incorrectly decoded.

<u>Derivation of decoding algorithm</u>: The number of errors 'e' and erasures 'r' has to satisfy $2e+r \leq 7$. If e_k and r_k represents number of errors and erasures for row k, for k = 0 to 2, then $2e_k+r_k \leq 2$ for at least one k, for otherwise $2e+r \leq 7$ is violated. Step 1 attempts to correct row k and assigns correction number w_k equal to number of errors corrected; If row k contains upto 2 erasures and correctable then $w_k = 0$; If an uncorrectable pattern is detected, then $w_k = 3$.

Step 2 identifies two least reliable rows if at least one of w_j's is nonzero and and forms $\phi(z)$. Then it becomes possible to compute $S_{0,0}$ and consequently 3 syndromes for each row.

Now, $2e_k+r_k \leq 3$ for at least one row. Step 3 corrects rows having few more error patterns like 3 erasures or 1 error and a erasure can be corrected. Finally, a row is detected as having errors.

In step 4, second stage starts, processing S, obtained as 1–D DFT of columns of R'. Either a row having errors is detected earlier or can be determined from $S_{0,1}$ and $S_{0,2}$.

Errors–and–Erasures Decoding of (21, 7, 8) code

1) Obtain R' as 1–D DFT of rows of r. Correct for single errors or 2 erasures using $R'_{k,1}$, $R'_{k,2}$ and $R'_{k,4}$, and find w_k for k=0 to 2.

2) If $(w_0, w_1, w_2) = 0$ go to step 4. Else, compute row–parity vector p and last 2 components of 1–D DFT of p gives $S_{0,1}$ and $S_{0,2}$. Identify two least reliable rows and form $\phi(z)$. Compute $S_{0,0}$ using $\phi(z)$. If $S_{0,0} = 1$, invert components of p. Assign 0th column of R' to p.

3) Process $R'_{k,0}$, $R'_{k,1}$, $R'_{k,2}$ and $R'_{k,4}$ for k = 0 to 2, to correct rows having either 3 erasures or 1 erasure but detected as uncorrectable at step 1. Identify any rows which can not be corrected.

4) Obtain S by taking 1–D DFT of columns of R'. If none of the rows are detected as having errors at step 3, find $\phi(z)$ from $S_{1,0}$ and $S_{2,0}$ and use it determine all the syndromes by column recursion.

5) 2–D DFT of S gives e.

Example 5.3.2: Consider that a codeword 1 1 0 1 0 0 1/1 1 1 0 1 0 0/0 0 1 1 1 0 1 is transmitted. The DFT coefficients are in $GF(2^6)$ generated by (x^6+x+1) and α is a primitive 63rd root of unity. The locations of erasures (denoted by x) and errors are as shown below

0	1	0	x	0	0	0
0	0	1	x	0	0	0
0	x	0	0	0	0	0

and R' obtained by taking 1–D DFT of rows of r is given by

r\c	0	1	2	3	4	5	6
0	1	α^9	α^{18}	α^9	α^{36}	α^{36}	α^{18}
1	0	1	1	α^{45}	1	α^{54}	α^{27}
2	1	α^9	α^{18}	α^{45}	α^{36}	α^{54}	α^{27}

<u>First Stage decoding</u>: The 1, 2 and 4 components of row–0 syndromes, $\{\alpha^9, \alpha^{18}, \alpha^{36}\}$, are corrected on the basis of (7, 4, 3) code. It is easily checked that the erasure polynomial $(1 + \alpha^{27}z)$ does not satisfy it and therefore $w_0 = 3$. Similarly $w_1 = 3$ as row–1 syndromes $\{1, 1, 1\}$ does not satisfy erasure polynomial $(1 + \alpha^{45}z)$; $w_2 = 0$ as $\{\alpha^9, \alpha^{18}, \alpha^{36}\}$ satisfy erasure polynomial $(1 + \alpha^9z)$. Row–parity vector $p = \{1, 0, 1\}$ $\xrightarrow{\text{1–D DFT}} \{0, \alpha^{21}, \alpha^{42}\}$. Now $\phi(z) = (1 + \alpha^{42}z + \alpha^{21}z^2)$, and $S_{0,0}$ is computed to be 0. Therefore row–0 and row–1 syndrome are augmented respectively to $\{1, \alpha^9, \alpha^{18}, \alpha^{36}\}$ and $\{0, 1, 1, 1\}$. The connection polynomial are then found to be $(1 + \alpha^{36}z + \alpha^{36}z^2)$ for row–0 and $(1 + z + \alpha^{45}z^2)$ for row–1. The extended syndromes are given by $(1, \alpha^9, \alpha^{18}, \alpha^{27}, \alpha^{36}, \alpha^{45}, \alpha^{54})$ for row–0 and $(0, 1, 1, \alpha^9, 1, \alpha^{36}, \alpha^{18})$ for row–1. Finally the partially corrected R' becomes

r\c	0	1	2	3	4	5	6
0	0	0	0	α^{36}	0	α^{18}	α^9
1	0	0	0	1	0	1	1
2	0	0	0	α^{54}	0	α^{27}	α^{45}

By taking 1–D DFT of columns of R', S is obtained. Since all the syndromes are zero, we have $S = C$, and second stage decoding is not required for this case.

5.3.2 Decoding of $(21, 9, 8)$

This code has 2 more information bits than $(21, 7, 8)$ code for the same correction capability. For this code $e_{BCH} = 2$ which is less than $e_{2-D} = 3$. Although 2–D decoding is of type–2 the advantage of 2–D decoding algorithm is that the decoding procedure is simple for all cases except for one error configuration which requires extensive computation. The generator polynomial $g(x) = m_0(x)m_1(x)m_3(x)m_7(x)$ and the zeros of the code are shown in Figure 5.3.2.

<u>Derivation of decoding algorithm</u> : Step 1 passes 3 consecutive syndromes of row 0 through B–M algorithm which finds the correct $\Lambda(z)$ for the error configurations (o) and $\binom{0}{0}$ and detects when either of

r\c	0	1	2	3	4	5	6
0	Z			Z		Z	Z
1	Z	Z	Z		Z		
2	Z	Z	Z		Z		

Figure 5.3.2 Configuration of zeros of (21, 9, 8)

(Z represents zero locations)

the error configurations (o,o) and (o,o,o) occur [4]. Step 2 finds the correct $\Lambda(z)$ for the error configurations $\binom{o}{o},o)$ and$\binom{o}{o})$.

Step 3 distinguishes between (o,o) and (o,o,o) based on whether the number of nonzero components of r is odd or even. In the case of (o,o) two column errors are either in the same row or in different rows. Consequently (o,o) is distinguished by either 0 or 2 nonzero components of p. $\phi(z)$ is found either from row locators having odd–parity or from the syndromes of column 1 and utilized to determine all the syndromes.

Step 4 handles the difficult case of the error configuration (o,o,o). We could not find an algebraic method to compute $\Lambda(z)$. We adapt trial–and–error method of estimating $S_{0,1} \epsilon \, GF(2^3)$ and find $\Lambda(z)$ having degree 3 which passes validity tests. The following lemma shows that for the error configuration of interest only one choice succeeds.

Lemma 5.3.1: Consider a (21, 9, 8) code with the zero locations as in Figure 5.3.2. For an error e with the configuration (o,o,o) there exists unique $S_{0,1} \epsilon \, GF(2^3)$ such that $\Lambda(z)$, computed by passing $S_{0,1}$ to $S_{0,6}$ through B–M algorithm, is of degree 3 and satisfies validity tests for all row syndromes.

Proof : If possible, let another $\Lambda'(z)$ also satisfy the requirements and e' be. the corresponding error pattern. Then both e and e' have the same syndrome. The claim is that the configuration of e' has to be (o,o,o). Since the $\Lambda(z)$ computed from $S_{0,1}$ to $S_{0,6}$ is of degree 3, it follows that column error pattern is either $\binom{o}{o}$ or (o). Note that there cannot be even weight column as $\Lambda'(z)$, like $\Lambda(z)$, has to satisfy all the row syndromes. There cannot be two columns with the error configurations $\binom{o}{o}$ as then, in step 2 row–2 syndromes satisfy connection polynomial of degree 1 which contradicts the hypothesis. If one of the column errors has the configuration $\binom{o}{o}$, then $\binom{o}{o},o,o)$ could well be e'.

We now show that although e $\oplus$ e' is of Hamming weight 8, it cannot be a codeword. The zero structure of (21, 9, 8) implies that $c'_{k,0}$, $c'_{k,1}$ and $c'_{k,2}$ for k = 0 to 2, are same for all rows, where $c'_{k,i}$ denotes i^{th} component of the 1–D DFT of kth row vector of the codeword c. It is seen that e $\oplus$ e' has 2

rows of the form $y^j \oplus y^m$ $y^j \oplus y^n$, where m ≠ n (This is because e has 3 errors in different columns) and j is the location of column error $\binom{8}{0}$. But if m ≠ n, then the equality of $c^!_{k,0}$, $c^!_{k,1}$ and $c^!_{k,2}$ cannot be satisfied and e ⊕ e' is not a codeword.

Therefore the configuration for e' has to be (0,0,0). But then weight of (e⊕e') is at most 6 which cannot be a codeword. Hence $S_{0,1}$ can be uniquely chosen. ☐

Decoding Algorithm for (21, 9, 8)

1) Pass $S_{0,5}$ to $S_{0,0}$ through B–M algorithm and find $\Lambda(z)$. If B–M algorithm detects a condition of occurrence of more than 1 error go to step 3. Else, complete recursion of row–0.

2) Update $\Lambda(z)$ by passing $S_{1,0}$ to $S_{1,2}$ through B–M algorithm. Complete recursion of rows 1 and 2. Go to step 5.

3) Compute the number of nonzero components (nrp) of row parity–check vector **p**.
   If nrp = 1 or 3, go to step 4.
   If nrp = 2, then form $\phi_{er}(z)$ having as roots locators of rows with odd parity–checks. Compute $S_{0,1}$, $S_{0,2}$ and $S_{0,4}$ using $\phi_{er}(z)$. Find $\Lambda(z)$ using $S_{0,0}$ to $S_{0,6}$. Compute the remaining syndromes of rows 1 and 2 using $\Lambda(z)$. Go to step 5.
   If nrp = 0, find $\phi(z)$ using $S_{1,1}$ and $S_{2,1}$. Find the remaining syndromes using $\phi(z)$. Go to step 5.

4) Find $S_{0,1} \in GF(2^3)$ such that the $\Lambda(z)$ computed by passing through $S_{0,1}$ to $S_{0,6}$ has degree 3 and satisfies the syndromes of row–1. Determine the syndromes row–2 using $\Lambda(z)$ by row recursive extension.

5) 2–D IDFT S gives e.

Example 5.3.3: Consider that a codeword 1 1 0 1 0 0 1/1 1 1 0 1 0 0/0 0 1 1 1 0 1 is transmitted. The DFT coefficients are in $GF(2^6)$ generated by (x^6+x+1) and α is a primitive 63rd root of unity. For $e(x,y) = (y^5 +xy+ x^2 y)$. The available syndromes are as shown below in bold and italicized entries.

r\c	0	1	2	3	4	5	6
0	1	α^{27}	α^{54}	1	α^{45}	1	1
1	0	α^{16}	α^{4}	α^{26}	α	α^{41}	α^{38}
2	0	α^{2}	α^{32}	α^{19}	α^{8}	α^{13}	α^{52}

and **p** = (1, 1, 1). In step 1, row–0 syndromes compute $\Lambda(z) = (1 +z)$, which is updated in step 2 to $(1 +\alpha^{51}z +\alpha^{53}z^2)$. Then also row–1 syndromes are not satisfied. At step 3 nrp = 3, and therefore step 4 has

to be performed. For the only choice of $S_{0,1} = \alpha^{27}$, B–M algorithm on row–0 syndromes produces $\Lambda(z) = (1 + \alpha^{27}z + \alpha^9 z^2 + \alpha^9 z^3)$ which satisfies the syndromes rows 1 and 2. Corresponding to this $\Lambda(z)$, the recursively extended syndromes are shown above in entries which are non–bold and non–italicized.

5.4 Decoding algorithms for codes of length 33

5.4.1 Decoding of $(33, 13, 10)$

This code has $e_{BCH} = 2 < e_{alg} = e_{2-D} = 4$. 2–D decoding algorithm is of type–1. The $g(x)$ for the code is given by $m_1(x)m_3(x)$ and the zeros are as shown in Figure 5.4.1.

r/c	0	1	2	3	4	5	6	7	8	9	10
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1		Z		Z	Z	Z				Z	
2			Z				Z	Z	Z		Z

Figure 5.4.1 Configuration of zeros of (33, 13, 10)

<u>Derivation of decoding algorithm</u> : Decoding starts with finding $\Lambda(z)$ using B–M algorithm on $S_{0,1}$ to $S_{0,10}$. For correcting error configurations (o, o, o, o) and (o, o, o) the knowledge of $\Lambda(z)$ is enough. Step 2 takes care of recursive extension for these two error configurations.

If $\deg(\Lambda(z)) = 2$ then different error configurations possible are $(o,o,{}^o_o)$ or (o,o) or $(o,{}^o_o)$. We use masking technique to obtain the information on the even weight column error locations. The idea is to use the known $\Lambda(z)$ as a erasure polynomial and obtain modified syndromes as in Forney's algorithm [31]. Rewriting $S_{i,j}$ as

$$S_{i,j} = \sum_{m=1}^{r} e_m(\alpha^i)Y_m^j + \sum_{n=r+1}^{t} e_n(\alpha^i)Y_n^j \qquad (5.4.1)$$

where first r positions refer to erasure locations and the next (t–r) locations refer to error locations and $\beta^{jk} = Y_k$ for $k = 1$ to t. Let $\Lambda_{er}(z) = \prod_{m=1}^{r}(1 + zY_m) = \sum_{k=0}^{r} \sigma_k z^k$ be the erasure polynomial. Define modified syndrome corresponding to $S_{i,j}$ as

$$S'_{i,j} = \sum_{k=0}^{r} S_{i,j-k}\sigma_k \qquad (5.4.2)$$

Then (5.4.1) can be rewritten as

$$S'_{i,j} = \sum_{m=1}^{r} e_m(\alpha^i)\sigma(Y_m^{-1}) Y_m^j + \sum_{n=r+1}^{t} e_n(\alpha^i)\sigma(Y_n^{-1}) Y_n^j \qquad (5.4.3)$$

As $\sigma(Y_m^{-1}) = 0$ for m=1 to r, this simplifies to

$$S^!_{i,j} = \sum_{n=r+1}^{t} e_n(\alpha^i)\sigma(Y_n^{-1})Y_n^j \tag{5.4.4}$$

Note that $S^!_{i,j}$ has information only from error locations. Unfortunately, this information on the error locations is at the expense of conjugacy property of modified syndromes. This is immediately seen from weighting factors $\sigma(Y_j^{-1})$ for j=t+1 to r, in (5.4.3). Consequently masking technique is of limited applicability when t+1 = r. or when only error location is to be known and, that too, subject to the condition that it should be possible to separate weighting factor. It is clear from Fig. 5.4.1 that $S^!_{1,5}$ and $S^!_{2,8}$ can be determined. For the (33, 13, 10) code it is possible to eliminate weighting factor as follows.

$$S^!_{2,8} = e(\alpha^2)\sigma(Y_e^{-1})Y_e^8; \qquad S^!_{1,5} = e(\alpha)\sigma(Y_e^{-1})Y_e^5$$
$$\sigma(Y_e^{-1}) = (S^!_{2,8})^2/S^!_{1,5}$$

Note that $S^!_{1,5} \neq 0$, for otherwise there would be no more errors. We obtain

$$S^{!!}_{1,5} = (\sigma(Y_e^{-1}))^{-1}S^!_{1,5} \qquad S^{!!}_{2,8} = (\sigma(Y_e^{-1}))^{-1}S^!_{2,8} \tag{5.4.5}$$

from which we can find other conjugate terms which help in computing $\Lambda'(z)$. The final $\Lambda(z)$ is updated as product of $\Lambda_{old}(z)\Lambda'(z)$. This explains step 3.

In step 4, $\deg(\Lambda(z)) = 1$, and $\Lambda(z)$ is updated by passing $S_{1,3}$ to $S_{1,5}$ through B–M algorithm. Then the recursive extensions are direct.

If the $\deg(\Lambda(z)) = 0$, then different possible configurations are $\binom{o}{o}$ and $\binom{o\ o}{o\,o}$. For this case the syndrome sets $\{S_{1,3},\ S_{1,4},\ S_{1,5}\}$ and $\{S_{2,6},\ S_{2,7},\ S_{2,8}\}$ have to be solved to find $\Lambda(z)$. The following lemma shows that for the configuration $\binom{o\ o}{o\,o}$, $\Lambda(z)$ can always be found by showing that the matrix of the system of equations is invertible.

Lemma 5.4.1 : For the error configuration $\binom{o\ o}{o\,o}$, the $\det(A) \neq 0$, where

$$A = \begin{bmatrix} S_{1,3} & S_{1,4} \\ S_{2,6} & S_{2,7} \end{bmatrix}$$

Proof: Assuming $e(x,y) = y^{j_1}e_1(x) + y^{j_2}e_2(x)$, we factor A as

$$A = \begin{bmatrix} e_1(\alpha) & e_2(\alpha) \\ Y_1^3 e_1(\alpha^2) & Y_2^3 e_2(\alpha^2) \end{bmatrix} \begin{bmatrix} Y_1^3 & Y_1^4 \\ Y_2^3 & Y_2^4 \end{bmatrix}$$

As the latter matrix is invertible Vandermonde matrix, $\det(A) \neq 0$, iff

$$Y_1^3 f(\alpha) + Y_2^3 f(\alpha^2) \neq 0$$

If $f(\alpha) = 1$, then we are through as $Y_1 \neq Y_2$. Taking $f(\alpha) = \alpha^k$ and $j' = j_1 - j_2$, it is sufficient to show that

$$\beta^{3j'} = \alpha^k$$

is impossible for nontrivial j' and k. As $\beta = \nu^3$ and $\alpha = \nu^{11}$ this implies the congruence equation

$$9j' \equiv 11k \pmod{33}$$

which does not have nontrivial solution in j' and k as $\gcd(9,33) = 3$ does not divide 11 [88]. $\square$

Decoding Algorithm for (33, 13, 10)

1) Pass $\{S_{0,j}\}_{j=1}^{10}$ through B–M algorithm and compute $\Lambda(z)$. Compute $S_{0,0}$ by recursive extension.

2) If $\deg(\Lambda(z)) = 4$, then

If $\Lambda_1 \neq 0$, determine unknown $S_{2,9}$ among the syndromes $S_{2,6}$ to $S_{2,10}$ and complete the recursion of rows 2 and 1; Go to step 6.

If $\Lambda_1 = 0$, determine $S_{1,7}$ from the syndromes $S_{1,3}$ to $S_{1,5}$. Compute $S_{1,6} = (S_{1,7})^4$. Determine all the syndromes of row 1 and then of row 1 using conjugacy relation. Go to step 5.

If $\deg(\Lambda(z)) = 3$, use $S_{1,3}$ to $S_{1,5}$ for recursive extension of first row and $S_{2,6}$ to $S_{2,8}$ for second row. Go to step 6.

3) If $\deg(\Lambda(z)) = 2$, obtain modified syndromes $S_{1,5}'$ and $S_{2,8}'$ as in (5.4.2). Compute $S_{1,5}'' = (\sigma(Y_t^{-1}))^{-1} S_{1,5}'$, $S_{2,8}'' = (\sigma(Y_t^{-1}))^{-1} S_{2,8}'$, $S_{2,7}'' = (S_{2,8}'')^4$ and $S_{2,7}'' = (S_{1,5}'')^8$. Find $\Lambda_1(z)$ from $S_{2,6}''$ to $S_{2,8}''$ and update $\Lambda(z) \longrightarrow \Lambda(z)\Lambda_1(z)$. Complete the recursion of rows 1 and 2. Go to step 6.

4) If $\deg(\Lambda(z)) = 1$, update $\Lambda(z)$ by passing $S_{1,3}$ to $S_{1,5}$ through B–M algorithm. Complete recursive extensions of rows 1 and 2 using $\Lambda(z)$. Go to step 6.

5) If $\deg(\Lambda(z)) = 0$, then obtain $\Lambda(z)$ by solving $\{S_{1,3}, S_{1,4}, S_{1,5}\}$ and $\{S_{2,6}, S_{1,7}, S_{1,8}\}$. Complete recursion of rows 1 and 2 using $\Lambda(z)$.

6) 2–D IDFT of S gives e.

Example 5.4.1: Consider the transmitted vector $(1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,0\,/\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0$ $/\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1)$. The 2–D DFT domain is $GF(2^{10})$ generated by $(x^{10}+x^3+1)$ with ν as 1023rd root of unity. For $e(x,y) = (y^3 + xy^4 + (x+x^2)y^6)$, the available syndromes are shown in bold and italicized entries.

r/c	0	1	2	3	4	5	6	7	8	9	10
0	0	ν^{936}	ν^{849}	ν^{234}	ν^{675}	ν^{654}	ν^{468}	ν^{117}	ν^{327}	ν^{570}	ν^{285}
1	ν^{341}	ν^{693}	ν^{910}	ν^{429}	ν^{726}	ν^{858}	ν^{739}	ν^{952}	ν^{571}	ν^{363}	ν^{238}
2	ν^{682}	ν^{455}	ν^{363}	ν^{881}	ν^{797}	ν^{119}	ν^{858}	ν^{726}	ν^{429}	ν^{476}	ν^{693}

Step 1 by passing row–0 syndromes through B–M algorithm produces $\Lambda(z) = (1 + \nu^{936}z + \nu^{651}z^2)$ and by recursive extension $S_{0,0} = 0$. So step 3 has to be performed and compute $S^!_{1,5} = \nu^{990}$ and $S^!_{2,8} = \nu^{618}$ and find $\sigma(Y_e^{-1}) = \nu^{1236-990} = \nu^{246}$. Therefore $S''_{1,5} = \nu^{744}$ and $S''_{2,8} = \nu^{372}$ and $\Lambda'(z)$ is found to be $(1 + \nu^{558}z)$ and $\Lambda(z)$ is updated to $(1 + \nu^{536}z + \nu^{970}z^3 + \nu^{186}z^3)$. Then recursively extended syndromes are shown above by entries which are non–bold and non–italicized. Once S is known, $C = S \oplus R$, and 2–D IDFT of C gives c.

5.4.2 Decoding of (33, 11, 11)

For this code $e_{BCH} = 3$ and $e_{alg} = 4$, both lower than $e_{2-D} = 5$. 2–D spectral decoding algorithm is of type–2. The (33, 10, 12) code is the even weight subcode of the (33, 11, 11) code. As (33, 10, 12) cannot correct more errors than (33, 11, 11) and the availability of $S_{0,0}$ does not lead to any simplifications, we do not give a separate decoding algorithm for the latter. The g(x) for (33, 11, 11) code is $m_1(x)m_3(x)m_{11}(x)$ and the zero locations are shown in Figure 5.4.2.

r/c	0	1	2	3	4	5	6	7	8	9	10
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z	Z		Z	Z	Z				Z	
2	Z		Z				Z	Z	Z		Z

Figure 5.4.2 Configuration of zeros of (33, 11, 11)

<u>Derivation of decoding algorithm</u> : Decoding starts with finding $\Lambda(z)$ using B–M algorithm on $S_{0,1}$ to $S_{0,10}$. For correcting error configurations (o, o, o, o, o) and (o, o, o, o) the knowledge of $\Lambda(z)$ is enough. Step 2 takes care of recursive extension for these two error configurations. If $\deg(\Lambda(z)) \leq 3$, the column errors are identified as follows. First, $C_{0,0}$ is computed as $S_{0,0} \oplus R_{0,0}$. If $C_{0,0} = 0$ then odd parity–check columns are in error; if $C_{0,0} = 1$, then even parity–check columns are in error.

In step 3 the error configurations considered are (o, o, o) and $(o, o, o, \begin{smallmatrix} o \\ o \end{smallmatrix})$. The procedure we adopt is to add a root to $\Lambda(z)$ from the set of non–affected column locators and this modified connection polynomial is checked for validity. The justification in this case for the uniqueness of $\Lambda(z)$ is straightforward using 2–D complexity theorem. This is the most computation intensive step of the algorithm.

If $\deg(\Lambda(z)) = 2$, then the case is handled as in the case of step 3 of decoding algorithm for (33, 13, 10).

If $\deg(\Lambda(z)) = 1$ then different error configurations are $(o, \begin{smallmatrix} o & o \\ o & o \end{smallmatrix})$ or $(o, \begin{smallmatrix} o \\ o \end{smallmatrix})$ or $(\begin{smallmatrix} 8 & o \\ 8 & o \end{smallmatrix})$ or (o). Since the column locator of the odd weight error is known, we try to guess the row position of the error. For each guess adapt the syndromes. Then from the syndrome sets $\{S_{1,3}, S_{1,4}, S_{1,5}\}$ and $\{S_{2,6}, S_{2,7}, S_{2,8}\}$ obtain $\Lambda(z)$ and test for ascertaining its validity. If an error is introduced in the wrong place, then the original e is modified to e' having 3 even weight columns. In such a case $\Lambda(z)$, which can be of degree at most 2, cannot satisfy validity tests. Because if it satisfies this would imply that $e' \oplus e''$ is a codeword where e'' is the codeword output by the 2–D decoder. Since e'' can have 3 even weight column errors, Hamming weight of $(e' \oplus e'')$ can be at most 10 which cannot be a codeword. This means that only one guess for row position passes the validity tests.

<div align="center">Decoding Algorithm for (33, 11, 11)</div>

1) Perform B–M algorithm on the syndromes $S_{0,1}$ to $S_{0,10}$ and obtain $\Lambda(z)$. Compute $S_{0,0}$ by recursion. If $\deg(\Lambda(z)) \leq 3$, compute $C_{0,0} = R_{0,0} \oplus S_{0,0}$. If $C_{0,0} = 0$, then identify the columns having odd–parity check as affected by error; else if $C_{0,0} = 1$, then identify even–parity check columns as containing errors.

2) If $\deg(\Lambda(z)) = 5$, then
If $\Lambda_2 \neq 0$ determine unknown $S_{2,9}$ among the syndromes $S_{2,6}$ to $S_{2,0}$ and complete the recursion of row 2 and then row 1; Go to step 8.
If $\Lambda_2 = 0$, first determine unknown $S_{1,10}$ among the syndromes $S_{1,0}$ to $S_{1,4}$ (This is possible since $\Lambda_5 \neq 0$). Determine $S_{2,9} = (S_{1,10})^2$. Complete the recursion of rows 2 and then 1. Go to step 8.

3) If $\deg(\Lambda(z)) = 4$, then
If $\Lambda_1 \neq 0$, determine unknown $S_{2,9}$ among the syndromes $S_{2,6}$ to $S_{2,10}$ and complete the recursion of rows 2 and 1; Go to step 8.
If $\Lambda_1 = 0$, determine $S_{1,7}$ from the syndromes $S_{1,3}$ to $S_{1,5}$. Compute $S_{1,6} = (S_{1,7})^4$. Determine all the syndromes of row 2 and then row 1 by using recursion. Go to step 8.

4) If $\deg(\Lambda(z)) = 3$, then find $j \in \{$ columns not in error$\}$ such that $\Lambda_1(z) = (1 + \alpha^j z)\Lambda(z)$ satisfies the syndromes of rows 1 and 2 and thus determine all the syndromes. Go to step 8.

5) If $\deg(\Lambda(z)) = 2$, obtain modified syndromes $S'_{1,5}$ and $S'_{2,8}$ $(S'_{i,j} = \sum_{k=0}^{2} S_{i,j-k}\sigma_k)$. Compute $\sigma(Y_e^{-1}) = (S'_{2,8})^2/S'_{1,5}$, if $S'_{1,5} \neq 0$; Determine $S''_{1,5} = (\sigma(Y_e^{-1}))^{-1}S'_{1,5}$, $S''_{2,8} = (\sigma(Y_e^{-1}))^{-1}S'_{2,8}$, $S''_{2,6} = (S''_{2,8})^4$ and $S''_{2,7} = (S'_{1,5})^8$. Find $\Lambda_1(z)$ from $S''_{2,6}$ to $S''_{2,8}$ and update $\Lambda(z) \longleftarrow \Lambda(z)\Lambda_1(z)$. Complete the recursion of rows 1 and 2. Go to step 8.

6) If $\deg(\Lambda(z)) = 1$, let 'j' is the column in error. For row position $i \in \{0,1,2\}$ do the following. Introduce an error at position (i,j). Adapt the syndromes. Obtain $\Lambda_1(z)$ by solving the system of equations formed by syndromes $\{S_{2,6}, S_{2,7}, S_{2,8}\}$ and $\{S_{1,3}, S_{1,4}, S_{1,5}\}$ and update $\Lambda(z) \longrightarrow \Lambda_1(z)$. Find $i \in \{0,1,2\}$ for which $\Lambda(z)$ passes validity tests for row 1 and row 2. and determine the the syndromes. Go to step 8.

7) If $\deg(\Lambda(z)) = 0$, solve the system of equations $\{S_{2,6}, S_{2,7}, S_{2,8}\}$ and $\{S_{1,3}, S_{1,4}, S_{1,5}\}$ to obtain $\Lambda(z)$. Complete the recursion of rows 1 and 2.

8) 2–D IDFT of S gives e.

Example 5.4.2: Consider the transmitted vector $(1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,0/\ /0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0$ $/0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1)$. The 2–D DFT is defined in $GF(2^{10})$ generated by $(x^{10}+x^3+1)$ and ν is 1023rd root of unity. For $e(x,y) = (1+x)y^2 +(x+x^2)y^7 +x^2y^9$, the available syndromes are as shown below

r/c	0	1	2	3	4	5	6	7	8	9	10
0		ν^{837}	ν^{651}	ν^{465}	ν^{279}	ν^{93}	ν^{930}	ν^{744}	ν^{558}	ν^{372}	ν^{186}
1		ν^{195}		ν^{816}	ν^{780}	ν^{51}				ν^{204}	
2			ν^{390}				ν^{609}	ν^{408}	ν^{537}		ν^{102}

Step 1 by passing row–0 syndromes through B–M algorithm produces $\Lambda(z) = (1 +\nu^{837}z)$ and by recursive extension $S_{0,0} = 1$. Since $\deg(\Lambda(z)) = 1$, step 6 has to be performed and the column location is 9. For errors at row–0 and row–1, $\Lambda(z)$ is found to be respectively, $(1 +\nu^{801}z +\nu^{744}z^2)$ and $(1 +\nu^{849}z +\nu^{279}z^2)$, but neither of them satisfies row–1 syndromes. Finally at row–2, $\Lambda(z) = (1 +\nu^{468}z +\nu^{837}z^2)$ found to satisfy row–1. Modify syndromes by correcting a single error at x^2y^9 and perform recursive extension of modified syndromes to get $(\nu^{341}, \nu^{182}, \nu^{980}, \nu^{557}, \nu^{728}, \nu^{866}, \nu^{245}, \nu^{317}, \nu^{851}, \nu^{395}, \nu^{335})$ for row–1 and

$(\nu^{682}, \nu^{490}, \nu^{364}, \nu^{634}, \nu^{937}, \nu^{679}, \nu^{91}, \nu^{790}, \nu^{433}, \nu^{670}, \nu^{709})$ for row–2. Once S is known, C = S⊕R, and 2–D IDFT of C gives c.

Example 5.4.3: Consider the transmitted code vector as in Example 5.4.2. For $e(x,y) = y^3 + xy^4 + (x+x^2)y^{10} + x^2y^9$, the available syndromes are shown below by bold and italicized entries.

r/c	0	1	2	3	4	5	6	7	8	9	10
0	1	ν^{672}	ν^{321}	ν^{168}	ν^{642}	ν^{522}	ν^{336}	ν^{84}	ν^{261}	ν^{42}	ν^{21}
1	1	ν^{918}	ν^{615}	ν^{741}	ν^{603}	ν^{366}	ν^{921}	ν^{486}	ν^{414}	ν^{441}	ν^{633}
2	1	ν^{819}	ν^{813}	ν^{972}	ν^{207}	ν^{828}	ν^{459}	ν^{882}	ν^{183}	ν^{243}	ν^{732}

Step 1 by passing row–0 syndromes through B–M algorithm produces $\Lambda(z) = (1 + \nu^{672}z + \nu^{486}z^2 + \nu^{465}z^3)$ and by recursive extension $S_{0,0} = 1$. The column errors are identified to be 3, 4 and 9. Since $\deg(\Lambda(z)) = 3$, step 4 has to be performed. It is seen that valid $\Lambda_1(z)$ is found for 10th column position equal to $(1 + \nu^{753}z + \nu^{120}z^2 + \nu^{939}z^3 + \nu^{372}z^4)$. The remaining syndromes are found using $\Lambda_1(z)$ are shown above in non–bold and non–italicized entries. Finally, C = S ⊕ R, and 2–D IDFT of C gives c.

5.5 Decoding of codes of length 35

5.5.1 Decoding of (35, 20, 6)

This code has $e_{\text{BCH}} = e_{2-D} = 2$ and 2–D decoding is of type–1. The advantages of 2–D decoding algorithm is that recursive extension lengths are small and that some more types of errors are correctable. The $g(x) = m_1(x)m_5(x)$ zero locations are shown in Figure 5.5.1.

r/c	0	1	2	3	4	5	6
0				Z		Z	Z
1		Z	Z		Z		
2		Z	Z		Z		
3		Z	Z		Z		
4		Z	Z		Z		

Figure 5.5.1 Configuration of zeros of (35, 20, 6)

2–D decoding (35, 20, 6) starts with the computation of the column connection polynomial $\phi(z)$. A typical row–error pattern is of the form $x^i g(y)$ and $g(\beta) = 0$ means that Hamming weight $g(y) \geq 3$. Therefore column–1 syndromes generate proper $\phi(z)$ if ≤ 2 random errors occur.

Decoding Algorithm for (35, 20, 6)

1) Compute $\phi(z)$ by passing $S_{1,1}$ to $S_{4,1}$ through B–M algorithm and use it to find $S_{0,1}$, $S_{0,2}$ and $S_{0,4}$ by column recursion.

2) Pass $S_{0,1}$ to $S_{0,6}$ through B–M algorithm and find $\Lambda(z)$. If $S_{0,1} = S_{0,3} = 0$, then find $\Lambda(z)$ from $S_{1,1}$ and $S_{1,2}$. Determine the rest of the syndromes by row recursion.

3) 2–D IDFT of S gives e.

5.5.2 Decoding of (35, 16, 7)

This code has $e_{BCH} = 2 < e_{alg} = e_{2-D}$. In [10] decoding algorithm makes use of Feng–Tzeng algorithm for multiple LFSR synthesis. The 2–D decoding algorithm is of type–1 and the advantage is that recursion lengths are small. 2–D decoding algorithm also uses Feng–Tzeng algorithm in a different way from [10]. The $g(x) = m_1(x)m_5(x)m_7(x)$ and the zero locations are as shown in Figure 5.5.2.

r/c	0	1	2	3	4	5	6
0				Z		Z	Z
1	Z	Z	Z		Z		
2	Z	Z	Z		Z		
3	Z	Z	Z		Z		
4	Z	Z	Z		Z		

Figure 5.5.2 Configuration of zeros of (35, 16, 7)

Derivation of decoding algorithm : The step 1 finds the correct $\Lambda(z)$ for the following column error configurations: (o), $\binom{o}{o}$, $\binom{o}{o}$, patterns of (o,o) where column errors are in different column, and $\binom{o}{o},o)$. Then the rest of the syndromes are easily computed using $\Lambda(z)$. The justification for step 1 is given by the following lemma.

Lemma 5.5.1: For $e(x,y) = y^{j_1}e_1(x) + y^{j_2}e_2(x)$, where $e_1(x) \neq e_2(x)$, $\det(A) \neq 0$, where

$$A = \begin{bmatrix} S_{1,0} & S_{1,1} \\ S_{2,0} & S_{2,1} \end{bmatrix}$$

Proof: Factoring A as

$$A = \begin{bmatrix} e_1(\alpha) & e_2(\alpha) \\ e_1(\alpha^2) & e_2(\alpha^2) \end{bmatrix} \begin{bmatrix} 1 & Y_1 \\ 1 & Y_2 \end{bmatrix}$$

Now $\det(A) = 0$, iff $e_1(\alpha)(e_2(\alpha))^2 = e_2(\alpha)(e_1(\alpha))^2$. This implies that

$$e_1(\alpha^k) = e_2(\alpha^k) \text{ for } k = 1 \text{ to } 4.$$

If maximum 3 errors occur, then Hamming weight of $(e_1(x)+e_2(x))$ is ≤ 3. By 2–D complexity theorem the spectrum of $e_1(x)-e_2(x)$ is all zero and therefore $e_1(x) = e_2(x)$ contradicting hypothesis. Hence the lemma is proved. □

If $\Lambda(z)$ fails validity test in step 1, a possible error configuration may be (o,o,o). We apply FT algorithm to syndrome sequences of columns 0,1 and 2 to obtain $\phi(z)$. This is the step 3 of the algorithm. The remaining steps after $\phi(z)$ are straightforward. The justification of step 2 is given in the following lemma.

Lemma 5.5.2: For the error configuration (o,o,o) FT algorithm on syndrome sequences of columns 0 to 2 produces unique $\phi(z)$.

Proof: Taking $e(x,y) = (x^{i_1}g_1(y) + x^{i_2}g_2(y) + x^{i_3}g_3(y))$ and assuming that $g_j(y) = \sum_{k=0}^{2} g_{j,k} y^{jk}$ for $j = 1$ to 3. Representing $X_k = \alpha^{ik}$ for $k = 1$ to 3 and $Y_m = \beta^{jm}$ for $j = 1$ to 3. We factorize the syndrome matrix as

$$\begin{bmatrix} S_{1,0} & S_{2,0} & S_{3,0} \\ S_{1,1} & S_{2,1} & S_{3,1} \\ S_{1,2} & S_{2,2} & S_{3,2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ Y_1 & Y_2 & Y_3 \\ Y_1^2 & Y_2^2 & Y_3^2 \end{bmatrix} \begin{bmatrix} g_{1,0} & g_{2,0} & g_{3,0} \\ g_{1,1} & g_{2,1} & g_{3,1} \\ g_{1,2} & g_{2,2} & g_{3,2} \end{bmatrix} \begin{bmatrix} X_1 & X_1^2 & X_1^3 \\ X_2 & X_2^2 & X_2^3 \\ X_3 & X_3^2 & X_3^3 \end{bmatrix}$$

$$A = CGR$$

As C and R are Vandermonde matrix, we have $\text{rank}(A) = \text{rank}(G)$. If 3 errors occur, then weight $g_k(x)$ is 1, for $k = 1$ to 3. If 1's are in different rows, then G is invertible and, as $S_{4,0}$, $S_{4,1}$ and $S_{4,2}$ are available, $\phi(z)$ can be computed. If two of $g_k(y)$'s are equal, say $g_1(y) = g_2(y)$, then $\text{rank}(G) = 2$ and the number of rows affected is also 2, therefore $\phi(z)$ can be computed. If $g_1(y) = g_2(y) = g_3(y)$, then $\text{rank}(G) = 1$, and the number of rows affected is also 1 and hence $\phi(z)$ can be computed. □

Decoding Algorithm for (35, 16, 7)

1) Solve $\{S_{1,0}, S_{1,1}, S_{1,2}\}$ and $\{S_{2,0}, S_{2,1}, S_{2,2}\}$ to obtain $\Lambda(z)$. Check whether $\Lambda(z)$ is satisfied by the syndromes of row–1. If yes, complete the recursion of all the rows. Go to step 4.

2) Using Feng–Tzeng algorithm on the 3 multiple sequence $\{S_{1,0}, S_{2,0}, S_{3,0}, S_{4,0}\}$ $\{S_{1,1}, S_{2,1}, S_{3,1}, S_{4,1}\}$ and $\{S_{1,2}, S_{2,2}, S_{3,2}, S_{4,2}\}$ and find $\phi(z)$. Using $\phi(z)$, find $S_{0,0}, S_{0,1}, S_{0,2}$ and $S_{0,4}$.

3) Pass $S_{0,1}$ to $S_{0,6}$ through B–M algorithm and find $\Lambda(z)$. Then find the remaining syndromes using $\Lambda(z)$.

4) 2–D IDFT of S gives e.

Example 5.5.1: The transmitted codeword is /1 1 0 0 0 1 0/ 0 1 0 1 0 0 1/ 0 1 0 1 0 0 1 /0 1 0 1 0 0 1/ 0 1 0 1 0 0 1. 2–D DFT is defined in $GF(2^{12})$ generated by $(x^{12}+x^{6}+x^{4}+x+1)$ and ν is primitive 4095th root of unity. If $e(x,y) = (xy + x^{2}y^{2} + x^{3}y^{3})$, the available syndromes, denoted by bold and italicized entries, are shown below

r/c	0	1	2	3	4	5	6
0	1	ν^{3510}	ν^{2925}	ν^{585}	ν^{1755}	ν^{2340}	ν^{1170}
1	ν^{3003}	ν^{691}	ν^{2866}	ν^{775}	ν^{811}	ν^{1840}	ν^{115}
2	ν^{1911}	ν^{1622}	ν^{1382}	ν^{3680}	ν^{1637}	ν^{230}	ν^{1550}
3	ν^{3549}	ν^{1433}	ν^{2453}	ν^{2105}	ν^{2393}	ν^{2435}	ν^{920}
4	ν^{3822}	ν^{3274}	ν^{3244}	ν^{460}	ν^{2764}	ν^{3100}	ν^{3265}

step 1 finds $\Lambda(z) = (1 + \nu^{1410}z + \nu^{2489}z^{2})$, which does not satisfy row–1 syndromes. FT algorithm, processing syndrome sequences of columns 0, 1 and 2, finds $\Lambda(z) = (1 + \nu^{3003}z + \nu^{546}z^{2} + \nu^{819}z^{3})$. The recursively extended entries of S are shown by entries which are of non–bold and non–italicized type. The calculation of c from R and S is straightforward.

5.5.3 Decoding of $(35, 15, 8)$

This is the even weight subcode of the code discussed above. The availability of the syndrome $S_{0,0}$ simplifies the computation of the $\phi(z)$. Decoding algorithm uses the fact that the roots of $\phi(z)$ are locators of columns having odd parity check. Consequently 2–D spectral decoding does not need to do Feng–Tzeng algorithm.

Decoding Algorithm for (35, 15, 8)

1) Compute the column parity checks for all columns and form $\phi(z)$ having as roots the locators of columns with odd parity. If the $\deg(\phi(z)) \leq 1$, update $\phi(z)$ by passing the syndromes of column 1 through B–M algorithm. Use $\phi(z)$ to determine $S_{0,1}$, $S_{0,2}$ and $S_{0,4}$.

2) Pass $S_{0,0}$ to $S_{0,6}$ through B–M algorithm and find $\Lambda(z)$. If the $\deg(\Lambda(z)) = 1$, update $\Lambda(z)$ by passing $S_{1,0}$ to $S_{1,2}$ through B–M algorithm. Find the rest of the syndromes using $\Lambda(z)$ by row recursion.

3) 2–D IDFT of S gives e.

5.5.4 Decoding of $(35, 7, 14)$

This code has $e_{BCH} = 5 < e_{2-D} = 6$. This code is the first example of 2–D codes that have type–3 decoding algorithm. The $g(x) = m_0(x)m_1(x)m_3(x)m_5(x)$ and the zeros are as shown in Figure 5.5.3.

r/c	0	1	2	3	4	5	6
0	Z			Z		Z	Z
1		Z	Z	Z	Z	Z	Z
2		Z	Z	Z	Z	Z	Z
3		Z	Z	Z	Z	Z	Z
4		Z	Z	Z	Z	Z	Z

Figure 5.5.3 Configuration of zeros of $(35, 7, 14)$

Derivation of decoding algorithm : The first stage of row decoding is based on $(7, 4, 3)$ which can correct single errors. The row decoding algorithm processes j^{th} row syndromes $R'_{j,k}$ at k = 3,4 and 5, for j = 0 to 4.

Single errors are corrected and the syndromes are adapted. A particular row decoding results in one of the following possibilities : a row error is either properly decoded or incorrectly decoded or not detected or detected as uncorrectable. If a row is incorrectly decoded then additional errors are added so that the row error becomes undetectable. With each corrected row attach a correction number w_j equal to number of corrected errors if decoder makes corrections; 3 if uncorrectable pattern is detected. A row with largest correction number is most unreliable.

In step 3, second stage of 2–D decoding algorithm processes S obtained by taking 1–D DFT of the columns of row–corrected R'. The row–wise configuration of errors not corrected after first stage are $\left(\begin{smallmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}8 & 8\\ 8 & 8\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0 & 0 & 0 & 0\\ & 0 & 0 &\end{smallmatrix}\right)$, (0,0,0,0,0) and (0,0,0,0,0,0). Except for $\left(\begin{smallmatrix}8 & 8\\ 8 & 8\end{smallmatrix}\right)$, in all the cases either at most 2 row errors or 1 row–erasure and 1 row–error can happen. Hence 4 consecutive syndromes of column 1 can generate valid $\phi(z)$. In the case of $\left(\begin{smallmatrix}8 & 8\\ 8 & 8\end{smallmatrix}\right)$ there can be x erasures and 3–x errors. Therefore $\phi(z)$ generated need not be valid. For the case when $\phi(z)$ produced by passing column syndrome sequence through B–M algorithm is invalid, then $\phi(z)$ can be formed from locators of uncorrectable rows and decoded rows having maximum correction numbers.

We invoke concatenated structure of 2–D cyclic codes to provide justification for the step 3. The nonzero locations of <0,1>, <0,2>, <0,4> can also be viewed as providing details of concatenated component of (35, 7, 14) code. In particular the concatenation is of the form I□O, where I is (7, 3, 4) over GF(2) and O is a (5, 1, 5) code over $GF(2^3)$. We have

$$d_{min}(I \,\square\, O) \geq d_{min}(O)\, d_{min}(I) = d_1 d_2 = 20$$

The first stage row–decoding can be viewed decoding of I. We have following theorem from [4].

Theorem 5.5.1: Let w_j be the correction number produced by row–decoder for j^{th} row of r. Let r' represent row–corrected r. Then there is exactly one column codeword satisfying

$$\sum_{j \in U} (d_2 - w_j) + \sum_{j \notin U} w_j \leq \lfloor (d_1 d_2 - 1)/2 \rfloor$$

where U is the set j for which the column codeword differs from the jth row of the r'.

Step 3 essentially applies errors and erasures decoding based on first stage decoding in the same way as explained in [4].

Once $\phi(z)$ is found the spread of error along columns in some cases when incorrect decoding happens can exceed 6. For example in the case of $\binom{8,8}{8,8}$ when 3 rows are incorrectly decoded the error may spread across all the 7 columns. Therefore additional errors introduced to incorrectly decoded rows are removed and S is adapted accordingly. Therefore $\Lambda(z)$ can be computed in a straightforward way and the remaining syndromes can be computed.

Decoding Algorithm for (35, 7, 14)

1) Obtain $\underline{R}_j^{'(r)}$ for $j = 0$ to 4, by taking 1–D DFT of rows of r. For rows $j = 0$ to 4, find $\Lambda_j(z)$ using $R'_{j,3}$, $R_{j,5}$ and $R_{j,6}$. If $\deg(\Lambda_j(z)) = 1$ and passes validity tests adapt R'. If an uncorrectable error pattern is detected leave a row as it is. Form $\phi_{uc}(z)$ having as roots the locators of rows for which uncorrectable error patterns are detected.

2) Compute S by taking 1–D DFT of columns of R'.

3) Compute $\phi(z)$ by passing $S_{1,0}$ to $S_{1,4}$ through B–M algorithm with the initialization $\phi_{uc}(z)$. If $\phi(z)$ is not valid then compute $\phi(z)$ as having 3 least reliable row locators as roots. Determine the syndromes $S_{0,1}$, $S_{0,2}$ and $S_{0,4}$ using $\phi(z)$. For rows identified as incorrectly decoded remove the errors introduced in step 1. Adapt the syndromes.

4) Form $\Lambda(z)$ by passing $S_{0,0}$ to $S_{0,6}$ through B–M algorithm. If $\deg(\Lambda(z)) \leq 4$, then update $\Lambda(z)$ by passing $S_{1,1}$ to $S_{1,6}$ through B–M algorithm. Find the remaining syndromes using $\Lambda(z)$.

5) 2–D DFT of the S gives e.

Example 5.5.2: The transmitted code vector is given as /1 1 0 0 0 1 0/ 0 0 1 1 1 0 1/ 1 1 0 0 0 1 0/ 1 1 0 0 0 1 0/ 1 1 0 0 0 1 0. 2–D DFT is defined over $GF(2^{12})$ as in Example 5.5.1. Suppose $e(x,y) = (y^3 + y^5) + x(1 + y^4) + x^2(y + y^6)$. Available 1–D DFT syndromes of for all the rows of r is shown below.

r/c	0	1	2	3	4	5	6
0				ν^{2340}		ν^{1170}	ν^{585}
1				ν^{2340}		ν^{1170}	ν^{585}
2				ν^{3510}		ν^{1755}	ν^{2925}
3				0		0	0
4				0		0	0

The first stage decoding based on $(7, 4, 3)$ code introduces an error at positions 6, 6 and 2, respectively for rows 0, 1 and 2. Therefore $w_0 = w_1 = w_2 = 1$ and $w_3 = w_4 = 0$. $\phi_{uc}(z) = 1$. From step 2 $S_{1,1}$ to $S_{1,4}$ is found to be $(\nu^{2326}, \nu^{3362}, \nu^{2228}, \nu^{1444})$. In step 3, $\phi(z)$ found from this sequence is $(1 + \nu^{2365}z + \nu^{2520}z^2)$ which does not pass the validity tests. Hence 3 least reliable rows 0, 1 and 2 are used to form $\phi(z) = (1 + \nu^{2184}z + \nu^{3003}z^2 + \nu^{2457}z^3)$. $S_{0,1}$, $S_{0,2}$ and $S_{0,4}$ are found by using $\phi(z)$ and S is shown below

r/c	0	1	2	3	4	5	6
0	0	0	0	ν^{3510}	0	ν^{1755}	ν^{2925}
1	0	ν^{3404}	ν^{1229}	ν^{1799}	ν^{3284}	ν^{1904}	ν^{119}
2	0	ν^{2473}	ν^{2713}	ν^{3808}	ν^{2458}	ν^{238}	ν^{3598}
3	0	ν^{2662}	ν^{1642}	ν^{2107}	ν^{1702}	ν^{2947}	ν^{952}
4	0	ν^{821}	ν^{851}	ν^{476}	ν^{1331}	ν^{3101}	ν^{3521}

Bold and italicized entries show syndromes known at the end of step 3. Step 4 computes $\Lambda(z)$ to be $(1 + \nu^{1170}z + \nu^{2340}z^2 + \nu^{3510}z^3 + \nu^{585}z^4 + \nu^{1755}z^5 + \nu^{2925}z^6)$ and non–bold and non–italicized entries represent syndromes obtained using recursion by $\Lambda(z)$.

5.6 Decoding Codes of length 39

5.6.1 Decoding of $(39, 15, 10)$

The $e_{BCH} = 3 < e_{2-D} = e_{alg} = 4$. 2–D decoding algorithm is of type −1. It is to be noted in this case decoding algorithm in [10] for correcting 4 errors is based on exhaustive computer search. The $g(x) = m_1(x)m_3(x)$ and the structure of the zero patterns is shown in Figure 5.6.1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1		Z		Z	Z					Z	Z		Z
2			Z			Z	Z	Z	Z			Z	

Figure 5.6.1 Configuration of zeros of (39, 15, 10)

Decoding Algorithm for (39, 15, 10)

1) Pass the $S_{0,1}$ to $S_{0,11}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{0,0}$ using $\Lambda(z)$.

2) If the $\deg(\Lambda(z)) \leq 2$ then update $\Lambda(z)$ by passing $S_{2,5}$ to $S_{2,8}$ through the B–M algorithm. Complete recursive extension of row 2.

3) Determine $S_{1,0} = (S_{2,0})^2$ and $S_{1,2} = (S_{2,1})^2$ and complete recursive extension of row 1.

4) 2–D IDFT of S gives e.

5.6.2 Decoding of (39, 13, 12)

This code has $e_{BCH} = 3 < e_{alg} = e_{2-D} = 5$. In this case decoding algorithm of [10] is based on exhaustive computer search. The zero locations are shown in Figure 5.6.2.below.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z	Z		Z	Z					Z	Z		Z
2	Z		Z			Z	Z	Z	Z			Z	

Figure 5.6.2 Configuration of zeros of (39, 13, 12)

Derivation of decoding algorithm : The decoding computation proceeds on the same lines as above except for the following error configurations $(o, o, o, \genfrac{}{}{0pt}{}{o}{o})$ and $(o, \genfrac{}{}{0pt}{}{o\,o}{o\,o})$. Correction of both these error patterns using B–M algorithm requires 5 consecutive syndromes with the knowledge of the locations of single errors. Masking technique does not work for this case. As in the case of (33, 11, 11) we use estimating $\Lambda(z)$ for the correction of $(o, o, o, \genfrac{}{}{0pt}{}{o}{o})$ and estimating row–error position for $(o, \genfrac{}{}{0pt}{}{o\,o}{o\,o})$,

Decoding Algorithm for (39, 13, 12)

1) Pass $S_{0,1}$ to $S_{0,11}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{0,0}$.

2) a) If the $\deg(\Lambda(z)) = 5$, and if $\Lambda_3 \neq 0$, then determine unknown $S_{1,11}$ among syndromes $S_{1,9}$ to $S_{1,1}$; else if $\Lambda_3 = 0$, determine $S_{1,5}$ using syndromes $S_{1,0}$ to $S_{1,4}$. Determine the rest of the syndromes of row 1. The syndromes of row 2 are found using conjugacy properties. Go to step 6.

b) If $\deg(\Lambda(z)) = 4$ then the recursive extension of rows 2 is straightforward. Then determine $S_{1,2}$ and use $S_{1,0}$ to $S_{1,4}$ to determine the rest of the syndromes of row 1. Go to step 6.

c) If $\deg(\Lambda(z)) = 2$, update $\Lambda(z)$ by passing $S_{2,5}$ to $S_{2,8}$ through B–M algorithm. Complete the recursion of rows 1 and 2 as above. Go to step 6.

3)   If $\deg(\Lambda(z)) = 3$, then find $j \in \{$ columns not in error$\}$ such that $\Lambda_1(z) = (1 + \alpha^j z)\, \Lambda(z)$ satisfies the syndromes of rows 1 and 2 and thus determine all the syndromes. Go to step 6.

4)   If $\deg(\Lambda(z)) = 1$, let 'j' is the column in error. For row position $i \in \{0,1,2\}$ do the following. Introduce an error at position (i,j). Adapt the syndromes. Obtain $\Lambda(z)$ by passing $S_{2,5}$ to $S_{2,8}$ through B–M algorithm. Find $i \in \{0, 1, 2\}$ for which $\Lambda(z)$ passes validity tests for row 1 and row 2. and determine the the syndromes. Go to step 6.

5)   If $\deg(\Lambda(z)) = 0$, obtain $\Lambda(z)$ by passing $S_{2,5}$ to $S_{2,8}$ through B–M algorithm. Determine the syndromes of rows 1 and 2.

6)   2–D IDFT of S gives e.

5.7 Decoding of codes of length 45

5.7.1. Decoding of $(45, 21, 8)$

For this code $e_{BCH} = e_{2-D}$. 2–D decoding is of type–1 and provides a good alternative. The $g(x) = m_1(x)m_5(x)m_9(x)m_{15}(x)$ and the structure of zero locations is as shown in Figure 5.7.1.

r/c	0	1	2	3	4	5	6	7	8
0		Z	Z	Z	Z	Z	Z	Z	Z
1	Z	Z			Z			Z	
2	Z		Z			Z			Z
3	Z		Z			Z			Z
4	Z	Z			Z			Z	

Figure 5.7.1 Configuration of zeros of $(45, 21, 8)$

In step 1, $\Lambda(z)$ is computed and $S_{0,0}$ is found. Then it becomes simple to find $\phi(z)$. Once $\phi(z)$ is known, syndromes of all the columns except 3 and 6 are determined. Once again updating $\Lambda(z)$ and performing recursive extension we determine all the syndromes.

Decoding Algorithm for (45, 21, 8)

1) Perform B–M algorithm on $S_{0,1}$ to $S_{0,8}$ to find $\Lambda(z)$ and compute $S_{0,0}$.

2) If $S_{0,0} = 0$, then compute row parity–check vector p and form $\phi(z)$ from locators of row j such that $p_j = 1$. If $S_{0,0} = 1$, then do likewise w.r.t. locators of $p_j = 0$. If the $\deg(\phi(z)) = 1$, update $\phi(z)$ by passing $S_{4,1}$ to $S_{1,1}$ through B–M algorithm. Complete the recursion of columns 1, 2, 4 and 5.

3) If the $\deg(\Lambda(z)) = 1$, then update $\Lambda(z)$ by passing $S_{1,0}$ to $S_{1,2}$ through B–M algorithm. Compute the remaining syndromes using $\Lambda(z)$.

4) 2–D IDFT of S gives e.

5.7.2 Decoding of (45, 15, 10)

This code has $e_{BCH} = 3 < e_{2-D} = 4$ and 2–D decoding is of type–2. For (45, 15, 10) code $g(x) = m_1(x)m_7(x)m_9(x)m_{15}(x)$ and the zero structure is as shown in Figure 5.7.2.

r/c	0	1	2	3	4	5	6	7	8
0				Z			Z		
1	Z	Z	Z		Z	Z		Z	Z
2	Z	Z	Z		Z	Z		Z	Z
3	Z	Z	Z		Z	Z		Z	Z
4	Z	Z	Z		Z	Z		Z	Z

Figure 5.7.2 Configuration of zeros of (45, 15, 10)

<u>Derivation of decoding algorithm</u> Step 1 constructs $\phi(z)$ based on locators of odd–parity rows and $\phi_{comp}(z)$ based on locators of even–parity rows. If at most 4 errors have occurred then only one of $\phi(z)$ or $\phi_{comp}(z)$ satisfies syndromes of row–1. This can be seen as follows. Let p be the row parity–check vector of the received vector r. Then row–parity check of e is p if $C_{0,0} = 0$ or p' if $S_{0,0} = 1$, where p' denotes bitwise complement of p. Since $p \oplus p' = (1, 1, 1, 1, 1)$, it follows that both p and p' have the same syndromes in $S_{0,1}$ to $S_{0,4}$. Therefore one of $\phi(z)$ or $\phi_{comp}(z)$ is valid. Using the valid connection polynomial it is straightforward to determine all the syndromes of row–0 by recursion. Then using $S_{0,0}$ to $S_{0,8}$ find $\Lambda(z)$ which can be used to find the remaining syndromes.

Decoding Algorithm for (45, 15, 10)

1) Form $\phi(z)$ from locators of all row j's such that $p_j = 1$ and $\phi_{comp}(z)$ from locators of row j's such that $p_j = 0$. If $\deg(\phi(z)) \leq 2$, then update $\phi(z)$ by passing $\{S_{k,1}\}_{k=1}^{4}$ through B–M algorithm. Complete the recursion of row 1 using $\phi(z)$ and test for validity. If $\phi(z)$ is not valid, repeat the process with $\phi_{comp}(z)$. Assign the valid connection polynomial to $\phi(z)$. Compute the syndromes in columns 1, 2, 4, 5, 7 and 8 using $\phi(z)$.

2) Find $\Lambda(z)$ passing $S_{0,0}$ to $S_{0,8}$ through B–M algorithm. If $\deg(\Lambda(z)) \leq 2$, then update $\Lambda(z)$ by passing $S_{1,7}$ to $S_{1,2}$ through B–M algorithm and find the remaining syndromes by row recursion.

3) 2–D IDFT of S gives e.

5.7.3 Decoding of (45, 14, 10)

This is the even–weight subcode of (45, 15, 10) code and has $e_{BCH} = 3$. The availability of $S_{0,0}$ permits simplification of decoding algorithm in step 1 compared to the previous case. The $g(x) = m_0(x)m_1(x)m_7(x)m_9(x)m_{15}(x)$ and the zero structure is as shown in Figure 5.7.3.

r/c	0	1	2	3	4	5	6	7	8
0	Z			Z			Z		
1	Z	Z	Z		Z	Z		Z	Z
2	Z	Z	Z		Z	Z		Z	Z
3	Z	Z	Z		Z	Z		Z	Z
4	Z	Z	Z		Z	Z		Z	Z

Figure 5.7.3 Configuration of zeros of (45, 14, 10)

Decoding Algorithm for (45, 14, 10)

1) Find the p and form $\phi(z)$ having as roots the row j's such that $p_j = 1$. If $\deg(\phi(z)) \leq 2$, then update $\phi(z)$ by passing $\{S_{k,1}\}_{k=1}^{4}$ through B–M algorithm. Compute the syndromes in columns 1, 2, 4, 5, 7 and 8 by column recursion.

2) Find $\Lambda(z)$ passing $S_{0,0}$ to $S_{0,8}$ through B–M algorithm. Update $\Lambda(z)$ by passing $S_{1,7}$ to $S_{1,2}$ through B–M algorithm. Find the remaining syndromes by row recursion.

3) 2–D IDFT of S gives e.

5.7.4 Decoding of $(45, 12, 10)$

This code has $e_{BCH} = 3 < e_{2-D} = 4$. 2-D decoding is of type-1. The $g(x) = m_0(x)m_1(x)m_3(x)m_7(x)m_9(x)$ and the zero locations are as shown in Figure 5.7.4.

r/c	0	1	2	3	4	5	6	7	8
0	Z								
1	Z	Z	Z		Z	Z	Z	Z	Z
2	Z	Z	Z	Z	Z	Z		Z	Z
3	Z	Z	Z	Z	Z	Z		Z	Z
4	Z	Z	Z		Z	Z	Z	Z	Z

Figure 5.7.4 Configuration of zeros of $(45, 12, 10)$

Decoding Algorithm for $(45, 12, 10)$ code

1) Pass $\{S_{1,j}\}_{j=4}^2$ through B-M algorithm and find $\Lambda(z)$. Compute the remaining syndromes in rows 1 to 4 using $\Lambda(z)$.

2) Compute p. Using locators row j's such that $p_j = 1$, form $\phi(z)$. Update $\phi(z)$ by passing $S_{1,1}$ to $S_{4,1}$ through B-M algorithm and, if need be, update again by passing through $S_{1,3}$ to $S_{4,3}$. Compute syndromes of row 0 by recursion.

3) 2-D IDFT of S gives e.

5.7.5 Decoding of $(45, 11, 9)$

This code $e_{BCH} = 3 < e_{2-D} = 4$. 2-D decoding is of type-1. The $g(x) = m_1(x)m_3(x)m_7(x)m_{15}(x)m_{21}(x)$ and the structure of zero locations is shown in Figure 5.7.5.

r/c	0	1	2	3	4	5	6	7	8
0				Z			Z		
1		Z	Z	Z	Z	Z	Z	Z	Z
2		Z	Z	Z	Z	Z	Z	Z	Z
3		Z	Z	Z	Z	Z	Z	Z	Z
4		Z	Z	Z	Z	Z	Z	Z	Z

Figure 5.7.5 Configuration of zeros of $(45, 11, 9)$

First step is direct. In step 2 computation of $\phi(z)$ is facilitated by taking 1-D IDFT of columns 3 and 6.

<div align="center">Decoding Algorithm for (45, 11, 9)</div>

1) Pass $\{S_{1,j}\}_{j=1}^{8}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{1,0}$ to $S_{4,0}$ by using $\Lambda(z)$ by row recursive extension.

2) Find $\underline{S}_{3}^{!(c)}$ obtained by taking 1–D IDFT of column 3. Using row locators corresponding to nonzero positions of $\underline{S}_{3}^{!(c)}$ form $\phi(z)$. If $\deg(\phi(z)) \leq 2$, update $\phi(z)$ by passing $S_{1,1}$ to $S_{4,1}$ through B–M algorithm. Update again for the column sequence $S_{1,0}$ to $S_{4,0}$ if $\deg(\phi(z)) \leq 2$. Compute the remaining syndromes of row 0 using $\phi(z)$.

3) 2–D IDFT of S to gives e.

5.7.6 Decoding of $(45, 9, 12)$ and $(45, 8, 12)$

The $e_{BCH} = 4 < e_{2-D} = 5$. 2–D decoding is of type–1. For (45, 9, 12), $g(x) = m_1(x)m_5(x)m_7(x)m_9(x)m_{15}(x)$ and the zero locations are shown in Figure 5.7.6.

r/c	0	1	2	3	4	5	6	7	8
0		Z	Z	Z	Z	Z	Z	Z	Z
1	Z	Z	Z		Z	Z		Z	Z
2	Z	Z	Z		Z	Z		Z	Z
3	Z	Z	Z		Z	Z		Z	Z
4	Z	Z	Z		Z	Z		Z	Z

<div align="center">Figure 5.7.6 Configuration of zeros of (45, 9, 12)</div>

<u>Derivation of decoding algorithm</u> : The step 1 determines $\Lambda(z)$ from row–0 syndromes uniquely if there are at most 3 columns having odd number of errors. If there are 4 or 5 columns having odd weight errors, then $\Lambda(z)$ is not uniquely determined. Because $S_{0,0} = 0$ or 1, construct two connection polynomials corresponding to either hypothesis. In fact these two are complement of each other, in the sense that if $\Lambda(z)$ has four column locators as roots then $\Lambda_{comp}(z)$ has remaining 5 column locators as roots. Now only one of $\Lambda(z)$ or $\Lambda_{comp}(z)$ satisfy syndromes of row–1 if at most 5 errors happened. The remaining steps are straightforward.

<div align="center">Decoding Algorithm for (45, 9, 12)</div>

1) Pass $S_{0,1}$ to $S_{0,8}$ through B–M algorithm to find $\Lambda(z)$.

If the $\deg(\Lambda(z)) \leq 3$ then compute $S_{0,0}$ and go to step 2.

If $\deg(\Lambda(z)) = 4$, construct $\Lambda_1(z)$ having complement locations as roots. Select one of $\Lambda(z)$ or $\Lambda_1(z)$,

whichever satisfies syndromes of row–1, and complete the recursion of all the rows. Go to step 3.

2)   If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{1,7}$ to $S_{1,2}$ through B–M algorithm. Then find the remaining syndromes by row recursion.

3)   2–D IDFT of S gives e.

For $(45, 8, 12)$ code $S_{0,0}$ is also available. This simplifies the step 1 to form $\Lambda(z)$ from the locators of columns with odd–parity. The remaining steps of the decoding algorithm remain as before.

Remark 5.7.1: Closer examination of $(45, 8, 12)$ shows that for codes of area 5 x n, having 2t consecutive zeros in row–0 and t consecutive zeros in any other row, error–correction capability is t. The proof of lower bound on minimum distance and the decoding algorithm are similar to the case of codes of area 3 x n considered Remark 5.3.2.

5.8 Decoding of codes of length 51

5.8.1 Decoding of $(51, 35, 5)$

All these codes have $e_{BCH} = 1 < e_{alg} = e_{2-D} = 2$. 2–D decoding is of type–1. For $(51, 35, 5)$ $g(x) = m_1(x)m_9(x)$ and the zero location are shown in Figure 5.8.1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0		Z	Z		Z				Z	Z				Z		Z	Z
1		Z			Z									Z			Z
2			Z						Z	Z					Z		

Figure 5.8.1 Configuration of zeros of $(51, 35, 5)$

<u>Derivation of Decoding Algorithm</u> : If $S_{0,1} = 0$ then the possible error configuration is $\binom{o}{o}$. Step 1 first checks for error configuration (o) by determining $\Lambda(z)$ determined from $S_{0,1}$ and $S_{0,2}$ and checking for validity. If not valid, then error configuration is (o,o) and this is handled by setting $S_{0,0} = 0$ and determining $\Lambda(z)$ from 5 consecutive syndromes. The remaining steps are straightforward.

Decoding Algorithm for (51, 35, 5)

1) If $S_{0,1} = 0$, go to step 2. Find $\Lambda(z)$ from $S_{0,1}$ to $S_{0,2}$ and check for validity. If not valid, set $S_{0,0} = 0$ and determine $\Lambda(z)$ from $S_{0,15}$ to $S_{0,2}$. If $\Lambda(z)$ is not satisfied, uncorrectable error condition is detected. Using $\Lambda(z)$ complete the determine the syndromes of all the rows. Go to step 3.

2) If the $\deg(\Lambda(z)) = 0$, then find $\Lambda(z)$ from $S_{2,8}$ to $S_{2,9}$ and perform recursive extension.

3) 2–D IDFT of S gives e.

5.8.2 Decoding of $(51, 34, 6)$–I and $(51, 34, 6)$–II

We discuss decoding algorithms for two codes having the parameters (51, 34,, 6). The (51, 34, 6)–I code is a even–weight of (51, 34, 5). The availability of $S_{0,0}$ simplifies step 1 as 5 consecutive zeros in row–0 are available. Hence the decoding computation is simple.

Decoding Algorithm for (51,34,6)–I

1) Pass $S_{0,15}$ to $S_{0,2}$ through B–M algorithm and find $\Lambda(z)$. Complete the recursive extension for all the rows. Go to step 3.

2) If the $\deg(\Lambda(z)) = 0$, then find $\Lambda(z)$ from $S_{2,8}$ to $S_{2,9}$ and perform recursive extension.

3) 2–D IDFT of S gives e.

Decoding of $(51, 34, 6)$–II

The 2–D decoding algorithm is of type–1 and slightly complex compared to algorithm for (51, 34, 6)–I and algorithm in [10] for (51, 34,6)–II. The reason for including this code is that it introduces a new idea. The $g(x) = m_0(x)m_1(x)m_5(x)$ and the zero locations are shown in Figure 5.8.2.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	Z																
1		Z			Z		Z	Z			Z	Z		Z			Z
2			Z	Z		Z			Z	Z			Z		Z	Z	

Figure 5.8.2 Configuration of zeros of (51, 34, 6)–II

Derivation of decoding algorithm : A look at the zero locations shows that it is not simple to compute $\Lambda(z)$ when two errors occur. The technique we adopt is to first obtain $\Lambda''(z)$ by first solving the syndrome sets $(S_{1,10}, S_{1,11}, S_{1,13})$ and $(S_{2,2}, S_{2,3}, S_{2,5})$. If the determinant of the system is not zero, then $\Lambda''(z)$ can be

expressed as $(1 + \sigma_1'' z^2 + \sigma_2'' z^3)$. Perform recursive extension using $\Lambda''(z)$. We now show that if $\Lambda''(z)$ generates periodic sequence then it implies that the corresponding $\Lambda(z)$ also generates periodic sequence. If $\Lambda(z) = (1 + \sigma_1 z + \sigma_2 z^2)$ then $\Lambda''(z) = (1 + (\sigma_1^2 + \sigma_2)z^2 + \sigma_1 \sigma_2 z^3) = \Lambda(z)(1 + \sigma_1 z)$. Therefore if $S(z)$ satisfies periodicity condition for $\Lambda''(z)$, we have

$$S(z)\Lambda''(z) \equiv 0 \pmod{z^{17}-1} \tag{5.8.1}$$

Since the determinant is not zero, $(1 + \sigma_1 z)S(z) \neq 0$ holds, and

$$S(z)\Lambda(z) \equiv 0 \pmod{z^{17}-1} \tag{5.8.2}$$

and therefore $S(z)$ also satisfies periodicity tests for $\Lambda''(z)$ iff it satisfies periodicity test for $\Lambda(z)$ if two errors happen.

Once $\Lambda''(z)$ is determined, all syndromes of rows 1 and 2 are determined. The 1–D IDFT of column 0 helps in finding $\phi(z)$ and the syndromes of row–0 is determined from column recursion. The following lemma shows that for error configuration of interest determinant of the system of equations in not zero.

Lemma 5.8.1 : For the error configuration (o,o), the det(A) $\neq$ 0, where

$$A = \begin{bmatrix} S_{1,10} & S_{1,11} \\ S_{2,2} & S_{2,3} \end{bmatrix}$$

Proof: Assuming $e(x,y) = y^{j_1}e_1(x) + y^{j_2}e_2(x)$, we factor A as

$$A = \begin{bmatrix} e_1(\alpha) & e_2(\alpha) \\ Y_1^9 e_1(\alpha^2) & Y_2^9 e_2(\alpha^2) \end{bmatrix} \begin{bmatrix} Y_1^{10} & Y_1^{11} \\ Y_2^{10} & Y_2^{11} \end{bmatrix}$$

where α is 3rd root of unity and $Y_k = \beta^{jk}$ for k= 1 to 2, where β is 17th root of unity. As the latter matrix is invertible Vandermonde matrix, det(A) $\neq$ 0, iff

$$Y_1^9 f(\alpha) + Y_2^9 f(\alpha^2) \neq 0$$

If $f(\alpha) = 1$, then we are through as $Y_1 \neq Y_2$. Taking $f(\alpha) = \alpha^k$ and $j' = j_1 - j_2$, it is sufficient to show that

$$\beta^{9j'} = \alpha^k$$

is impossible for nontrivial j' and k. As $\beta = \nu^3$ and $\alpha = \nu^{17}$, where ν is 51th root of unity, we have the congruence equation

$$27j' \equiv 17k \pmod{51}$$

which does not have nontrivial solution in j' and k as gcd(27 , 51) = 3 does not divide 17 [88]. $\square$

<u>Derivation of decoding algorithm</u> : It is evident from inspection of zero location is that decoding procedure is straightforward except for error configurations $(0,0,0)$ and $(0,{}^0_0)$. In the first case difficulty arises in performing recursive computation and in the latter case difficulty arises in finding the connection polynomial. The technique adopted is to estimate $S_{1,0} \in GF(2^2)$ and by trial–and–error method find $S_{1,0}$ which produces the valid $\Lambda(z)$ in both the cases. The uniqueness of choice is easily seen from 2–D complexity theorem. For the error configuration $e_1 = (0,0,0)$, e_2 is either $(0,0,0)$ or one or more columns having $({}^0_0)$ configuration. As $\Lambda(z)$ is the same in both cases it follows that $e_1 \oplus e_2$ can have weight at most 6 which cannot be a codeword. The uniqueness of choice for $(0,{}^0_0)$ can be proved similarly.

<div align="center">Decoding Algorithm for (51, 27, 8)</div>

1) Pass $\{S_{0,j}\}_{j=1}^{10}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{0,0}$ using $\Lambda(z)$.

2) If $\deg(\Lambda(z)) = 0$, then find the connection polynomial from $S_{2,8}$ and $S_{2,9}$.

3) If $\deg(\Lambda(z)) = 2$ or 0, then use recursive extension to find all the syndromes of row–2. Then use conjugacy property to know all the syndromes of row–1. Go to step 6.

4) If $\deg(\Lambda(z)) = 3$, then find $S_{1,0} \in GF(2^2)$, such that row–1 syndromes satisfies $\Lambda(z)$. Then compute the syndromes of row–2 using conjugacy property. Go to step 6.

5) If $\deg(\Lambda(z)) = 1$, then find $S_{1,0} \in GF(2^2)$, such that $\Lambda'(z)$ compute from $\{S_{1,16}, S_{1,0}, S_{1,1}\}$ after initializing with $\Lambda(z)$ satisfies the row–1 syndromes. Compute row–2 syndromes using conjugacy property.

6) 2–D IDFT of S gives e.

5.8.4 Decoding of $(51, 25, 8)$

Decoding is very much simplified if the code has $C_{1,0}$ and $C_{2,0}$ as zeros and then $g(x) = m_1(x)m_3(x)m_9()m_{17}(x)$. So the price paid in the previous case for having two extra information bits is the complex decoding algorithm.

<div align="center">Decoding Algorithm for (51, 25, 8)</div>

1) Pass $\{S_{0,j}\}_{j=1}^{10}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{0,0}$.

2) If $\deg(\Lambda(z)) = 3$, update $\Lambda(z)$ by passing $\{S_{1,16}, S_{1,0}, S_{1,1}\}$ through B–M algorithm. Compute all the syndromes of row –1 by row recursion and that of row–2 using conjugacy property.

3) 2–D IDFT of S gives e.

5.8.5 Decoding of $(51, 19, 10)$

This code has $e_{BCH} = 2 < e_{2-D} = 4$. 2–D decoding algorithm is of type–3 for $(51, 19, 12)$ simplifies to type–1 for $(51, 17, 12)$. The $g(x)$ for the code is $m_1(x)m_3(x)m_9(x)m_{19}(x)$ and the zero locations are shown in Figure 5.8.4.

<u>Derivation of decoding algorithm</u> : Notice that rows are codewords of $(17, 9, 5)$ code which can correct 2 errors. The decoding of $(17, 9, 5)$ is done as in [10]. Append to each row the number of errors corrected. After row–decoding is over, 1–D DFT of the columns of row–corrected array is taken. The row decoding is either correct or incorrect or uncorrectable. A row error cannot go undetected as this requires

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1		Z	Z		Z				Z	Z				Z		Z	Z
2		Z	Z		Z				Z	Z				Z		Z	Z

Figure 5.8.4 Configuration of zeros of $(51, 19, 10)$

at least 5 errors which is greater than 4, the error–correcting capability of the code.

If $S_{0,3} = 0$ then all errors are corrected. If $S_{0,3} \neq 0$, then there cannot be more than one uncorrectable row or incorrectly decoded row. If uncorrectable row is detected then the position of uncorrectable row is used to form $\phi(z)$ which is used for column recursion. If no uncorrectable row is detected then the incorrectly decoded row is identified by that row which has maximum errors corrected if at most 4 errors occurred. This can be seen as follows. For row–wise error–configuration $\binom{0}{0}$ the row–decoder outputs numbers 2 and 1. If we take the row that corrected 1 error as incorrectly decoded, then the number of errors $2+(5-1) = 6 > 4$. For the case of $\binom{0}{0}$ proof is similar.

Once $\phi(z)$ is known the remaining steps of the decoding algorithm are straightforward.

Decoding Algorithm for $(51, 19, 10)$

1) Compute $\underline{R}_k^{\prime(r)}$ for $k = 0$ to 2, by taking 1–D DFT of rows of r.

2) Decode each of rows $\{R_{k,j}^{\prime}\}_{j=0}^{16}$ for $k = 0$ to 2 according to algorithm given in [10]. Adapt the syndromes. Also note the number of errors corrected. For $\phi_{uc}(z)$ from rows detected as uncorrectable.

3) Compute S by taking 1–D DFT of columns of $\{R_{k,j}^{\prime}\}_{k=0}^{2}$ for $j = 0$ to 15. If $S_{0,3} = 0$, go to step 6.

4) If $\phi_{uc}(z) \neq 1$, then set $\phi(z) = \phi_{uc}(z)$. Else, identify the row for which maximum number of errors are corrected and form $\phi(z)$. Determine the remaining syndromes of columns 1 to 16 using $\phi(z)$ by column recursion.

5) Find $\Lambda(z)$ by passing $S_{0,1}$ to $S_{0,16}$ through B–M algorithm. Update $\Lambda(z)$ by passing $S_{1,1}$ to $S_{1,16}$ through B–M algorithm, if necessary. Find the remaining syndromes of row–0.

6) 2–D IDFT of S gives the e.

5.8.6 Decoding of $(51, 17, 12)$

This code also has $e_{BCH} = 2 < e_{2-D} = 5$. Compared to (51, 19, 10) $C_{1,0}$ and $C_{2,0}$ are additional zero locations, correction capability increased by 1 and 2–D BCH decoding algorithm is of type–1.

<div align="center">Decoding Algorithm for (51,17,12)</div>

1) Pass $\{S_{0,j}\}_{j=1}^{16}$ through B–M algorithm and find $\Lambda(z)$. Compute $S_{0,0}$ by recursion.

2) If $\deg(\Lambda(z)) \leq 3$, then update $\Lambda(z)$ by passing the sequence $S_{1,15}$ to $S_{1,2}$ through B–M algorithm. Complete the recursion using $\Lambda(z)$.

3) 2–D IDFT of S gives e.

5.8.7 Decoding of $(51, 11, 15)$

This code has $e_{BCH} = 4 < e_{2-D} = 7$. 2–D decoding is of type–3. The $g(x) = m_1(x)m_3(x)m_5(x)m_{11}(x)m_{19}(x)$ and the zero locations are shown in Figure 5.8.5.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0		Z	Z		Z				Z	Z				Z		Z	Z
1		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
2		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

<div align="center">Figure 5.8.5 Configuration of zeros of (51, 11, 15)</div>

<u>Derivation of decoding algorithm</u> : The first stage of row decoding is based on (17, 9, 5) which can correct upto 2 errors using decoding algorithm in [10]. Append to each row the correction number after first stage decoding. The row decoding is either correct or incorrect or uncorrectable. A row error may also go undetected.

Second stage of 2–D decoding processes S obtained by taking 1–D DFT of the columns of row–corrected $\{R'_{j,k}\}$. $S_{1,3} = S_{2,3} = 0$ is an indication that all errors are corrected. If $S_{0,3} \neq 0$, then

row—wise configuration of errors not corrected after first stage are $\left(\begin{smallmatrix}8,8\\0,0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}8,8\\8,0\\0\end{smallmatrix}\right)$, and a single row error having 5 or 6 errors. Therefore there can be either at most two row—errors (two incorrect rows) or one row—error and one row—erasure or two row—erasures. If there is only one row—error then the syndromes of column 1 generate valid $\phi(z)$. Otherwise, valid $\phi(z)$ can be formed from product of $\phi_{er}(z)$ and/or row locators of decoded rows having maximum correction numbers. Once $\phi(z)$ is found the syndromes of row—0 are determined.

If there are incorrectly decoded rows the additional errors added should be removed before computation of $\Lambda(z)$. Then $\Lambda(z)$ can be computed in a straightforward way.

Decoding Algorithm for (51, 11, 15)

1) Compute $\{R'_{k,j}\}_{j=0}^{16}$ for k = 0 to 2, by taking 1—D DFT of rows of r.

2) Decode each of rows $\{R'_{k,j}\}_{j=0}^{16}$ for k = 0 to 2, according to algorithm given in [10]. Adapt the syndromes. Note w_j the number of errors corrected for j^{th} row properly decoded. Form $\phi_{uc}(z)$ from rows detected as uncorrectable.

3) Compute S by taking 1—D DFT of column array $\{R'_{k,j}\}_{k=0}^{2}$ for j = 0 to 16. If $S_{1,3} = S_{2,3} = 0$, go to step 5.

4) If $\deg(\phi_{uc}(z)) = 2$, then set $\phi(z) = \phi_{uc}(z)$. Determine the remaining syndromes of row—0.

   If $\deg(\phi_{uc}(z)) = 0$, then find $\phi(z)$ from $S_{1,3}$ and $S_{2,3}$ and check for validity. If not valid, identify two corrected rows for which correction number is maximum and form $\phi(z)$. Determine the remaining syndromes of row—0.

   If $\deg(\phi_{uc}(z)) = 1$, then form $\phi(z)$ as product of $\phi_{uc}(z)$ and the row locator of least reliable row. Determine the remaining syndromes of row—0.

5) If there are incorrectly decoded rows remove the error introduced in step 1. Adapt the syndromes. Find $\Lambda(z)$ by passing $S_{0,1}$ to $S_{0,16}$ through B—M algorithm. Update $\Lambda(z)$ by passing through $S_{1,1}$ to $S_{1,16}$. Compute $S_{0,0}$, $S_{0,1}$ and $S_{0,2}$.

6) 2—D IDFT of S gives e.

5.9 Decoding of codes length 63

5.9.1 Decoding of $(63, 24, 16)$

In [56] Jensen has constructed a class of low rate cyclic codes of length $2^{2m-1}-1$ having more information bits than the corresponding dual of a high–rate BCH code of length $2^{2m-1}-1$, both codes correcting the same number of errors. Though this code can be decode using concatenated method of decoding, we show for a code that our methods can be applied. Unfortunately we could not obtain generalizations but it appears possible.

The majority of steps of the 2–D decoding algorithm is of type–1, but some steps are of type–3. The $g(x) = m_0(x)m_1(x)m_5(x)m_7(x)m_9(x)m_{15}(x)m_{23}(x)m_{21}(x)m_{27}(x)$ and the zero locations are as shown in Figure 5.9.1.

r/c	0	1	2	3	4	5	6	7	8
0	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z	Z	Z	Z			Z	Z	Z
2	Z		Z	Z	Z	Z	Z	Z	
3	Z	Z							Z
4	Z	Z		Z	Z	Z	Z		Z
5	Z				Z	Z			
6	Z		Z					Z	

Figure 5.9.1 Configuration of zeros of $(63, 24, 16)$

Derivation of decoding algorithm : Step 1 computes $\Lambda(z)$ by passing syndromes of row–0 through B–M algorithm. If necessary, $\Lambda(z)$ is updated by passing through syndromes of row–1 and in some cases by row–2. The step determines syndromes of rows 1, 2 and 4. For some cases all the syndromes of rows 3, 5 and 6 are found and in some other cases, for example column–wise configuration $\binom{0}{8},o,o,o)$, $\Lambda(z)$ may turn out to be not valid.

Second stage corrects only odd–error columns using 1–D IDFT of rows 0, 1, 2 and 4. Only column errors with 3 errors produce an incorrect pattern. The information on incorrectly decoded column can be known from $S_{3,8}$ to $S_{3,1}$. Then the rest of the syndromes are computed.

Decoding Algorithm for $(63, 24, 16)$

1)   Compute all the column parity–checks and form $\Lambda(z)$ corresponding to locators of columns of odd–parity.

If $\deg(\Lambda(z)) \leq 5$, update $\Lambda(z)$ by passing $S_{1,6}$ to $S_{1,3}$ through B–M algorithm. Determine syndromes in rows 1, 2 and 4 using $\Lambda(z)$. Further if $\deg(\Lambda(z)) \leq 3$, then complete the recursion of other rows and go to step 3.

If $\deg(\Lambda(z)) \leq 1$ even after two passes through B–M algorithm corresponding to rows 0 and 1, then update $\Lambda(z)$ by passing $S_{3,8}$, $S_{3,1}$ and $S_{3,2}$ through B–M algorithm. Complete row–recursion of all the columns. Go to step 3

2)   Compute the 1–D IDFT for rows 0,1,2 and 4. For columns of odd–parity perform single error correction or detect if more than single errors has happened. Adapt the syndromes. Form $\Lambda_{uc}(z)$ from uncorrectable columns. If $\Lambda_{uc}(z)=1$, then find $\Lambda(z)$ from $S_{3,8}$, $S_{3,0}$ and $S_{3,1}$. Determine the remaining syndromes by row recursion.

3)   2–D IDFT of S gives e.

5.9.2 Decoding of $(63, 39, 8)$ code

For this code Tzeng and Feng algorithm can be applied for 3 error–correction [53]. The 2–D decoding algorithm is of type–1. The $g(x) = m_1(x)m_3(x)m_{15}(x)m_{31}(x)$ and the zeros are shown in Figure 5.9.2.

r/c	0	1	2	3	4	5	6	7	8
0									
1		Z		Z			Z		Z
2			Z	Z			Z	Z	
3				Z	Z	Z	Z		
4				Z	Z	Z	Z		
5			Z	Z			Z	Z	
6		Z		Z			Z		Z

Figure 5.9.2 Configuration of zeros of $(63, 39, 8)$

Most of the steps are straightforward except for step 2 which handles some patterns of error having row–wise configuration $\binom{0}{0}, 0$. For some cases, column–3 syndromes produce the $\phi(z)$ of degree 1, whose root gives the location of the row containing a single error. Using 1–D IDFT of syndromes of columns 3 and 6, the column position of the error can be determined. After adapting syndromes for correction of single error, the new column connection polynomial is found from the syndromes of column 4 and the remaining syndromes are found.

Decoding Algorithm for (63, 39, 8)

1) Find the $\phi(z)$ by passing $S_{1,3}$ to $S_{6,3}$ through B–M algorithm and complete the recursion of columns 3 and 6.

2) a) If the $\deg(\phi(z)) = 1$ and check whether $\phi(z)$ satisfies $S_{3,4}$ and $S_{4,4}$. If it satisfies compute all the syndromes using $\phi(z)$. Go to step 3. Else, identify row in error, and correct the error by finding the 1–D IDFT of column 3 and 6. Adapt the syndromes. Find new $\phi(z)$ using $S_{3,4}$ and $S_{4,4}$ and use it to determine all the syndromes using $\phi(z)$.

b) If $\deg(\phi(z)) = 2$, then determine all the syndromes using column recursion. If the $\deg(\phi(z)) = 3$, then complete the recursion of columns 3 and 6. Find the roots of the $\phi(z)$ and identify rows in error. Correct the errors by finding the 1–D IDFT of column 3 and 6. Error correction is complete.

3) 2–D IDFT of S gives e.

Chapter 6

SPECTRAL DOMAIN DECODING ALGORITHMS
FOR 2-D CYCLIC CODES

This chapter presents application of DFT domain decoding methodology to decode various 2–D cyclic codes upto their error–correcting capability. Some good 2–D cyclic codes have been compiled in [56]. These are good in the sense that for a given block length and dimension they have the maximum possible distance among the class of linear codes. Hence obtaining decoding algorithms for such codes is of practical interest. We have been able to obtain good 2–D spectral decoding algorithms correcting upto full error–correcting capability for almost all important codes in the list of [56].

These 2–D codes, unlike 1–D codes discussed in Chapter 5, do not have 1–D characterization. Consequently, 1–D BCH decoding algorithms cannot be applied to decode them. In [56], Jensen has shown that 2–D cyclic codes can be interpreted as concatenated codes. This suggests that one can use concatenated structure for the purposes of decoding of 2–D cyclic codes [83]–[84]. Concatenated decoding, however, requires detailed knowledge of the encoding procedure of concatenated code. Here we show that it is possible to decode such codes in a different manner by application of spectral techniques, which operate on 2–D DFT of the received data.

Before taking up the details, we would like to comment on the relative complexities of concatenated decoding algorithms and 2–D spectral decoding algorithms discussed here. The classification of 2–D spectral decoding algorithms, considered in Chapter 5, into three types is based on the increasing computational and implementational complexities of 2–D decoding algorithms. We have seen that type–3 decoding algorithms are similar to concatenated decoding algorithms. This means that type–1 and type–2 algorithms represent situations where 2–D DFT characterization provides efficient decoding algorithms. Even for type–3 algorithms, the advantage of 2–D spectral techniques lies in not being tied to any particular method of encoding.

6.1 General outlines of 2-D spectral decoding algorithms for 2-D cyclic codes

The approach for designing 2–D cyclic decoding algorithm is no different from designing decoding algorithms for 1–D cyclic codes. The development of decoding algorithms for 2–D cyclic codes is similar to that of decoding algorithms for 1–D cyclic codes. The notations for explaining 2–D decoding algorithms remain as before. Jensen has complied good 2–D cyclic codes of length 45, 49 and 75 in [56]. Table 6.2.1 gives details of codes, taken from [56], for which 2–D decoding algorithms have been given here.

Table 6.2.1 2–D Cyclic Codes having 2–D Decoding Algorithms

Area of the code	Parameters of the code					Leaders of zero conjugacy classes
	n	k	d	e_{2-D}	Type	
3x15	45	25	8	3	1	$0_3,0_5,0_7,1_0,1_7,2_7$
	45	17	12	5	1	$0_1,0_3,0_5,0_7,1_1,1_2,1_3,1_5$
	45	15	14	6	1	$0_1,0_3,0_5,0_7,1_0,1_1,1_2,1_3,1_5$
	45^1	10	16	7	3	$0_0,0_1,0_3,0_7,1_0,1_3,1_5,1_7,2_3,2_5,2_7$
	45^2	10	16	7	1	$0_0,0_1,0_5,0_3,0_7,1_0,1_3,1_5,1_7,2_3,2_7$
	45	8	20	9	1	$0_0,0_1,0_3,0_5,0_7,1_1,1_3,1_5,1_7,2_1,2_3$
	45^1	6	22	10	3	$0_0,0_3,0_5,0_7,1_1,1_3,1_5,1_7,2_1,2_3,2_5,2_7$
	45^2	6	22	10	1	$0_0,0_1,0_3,0_5,0_7,1_3,1_5,1_7,2_1,2_3,2_5,2_7$
7x7	49	34	6	2	1	$0_1,0_3,1_0,3_0,3_6$
	49	30	8	3	1	$0_0,0_1,0_3,1_0,3_0,3_5,3_6$
	49	31	7	3	1	$0_1,0_3,1_0,3_0,3_5,3_6$
	49	21	12	5	3	$0_0,0_1,0_3,1_0,1_6,3_0,3_3,3_4,3_5,3_6$
	49	15	14	6	3	$0_0,1_0,1_4,1_5,1_6,3_0,3_1,3_2,3_3,3_4,3_5,3_6$
	49	13	16	7	3	$0_1,0_3,1_4,1_5,1_6,3_0,3_1,3_2,3_3,3_4,3_5,3_6$
	49	10	20	9	3	$0_1,0_3,1_0,1_4,1_5,1_6,3_0,3_1,3_2,3_3,3_4,3_5,3_6$
	49	6	24	11	3	$0_0,0_1,0_3,1_0,1_3,1_4,1_5,1_6,3_0,3_1,3_2,3_3,3_4,3_5,3_6$
5x15	75	58	6	2	1	$0_0,0_3,0_7,1_0,1_{14}$
	75	52	8	3	1	$0_0,0_3,0_5,0_7,1_0,1_{13},1_{14}$
	75	46	10	4	1	$0_0,0_1,0_3,0_7,1_0,1_{12},1_{13},1_{14}$
	75	41	12	5	1	$0_1,0_3,0_5,0_7,1_0,1_{11},1_{12},1_{13},1_{14}$
	75	37	14	6	1	$0_1,0_3,0_5,0_7,1_0,1_{10},1_{11},1_{12},1_{13},1_{14}$
	75	32	16	7	1	$0_0,0_1,0_3,0_5,0_7,1_0,1_9,1_{10},1_{11},1_{12},1_{13},1_{14}$
	75	28	18	8	1	$0_0,0_1,0_3,0_5,0_7,1_0,1_8,1_9,1_{10},1_{11},1_{12},1_{13},1_{14}$
	75	24	20	9	1	$0_0,0_1,0_3,0_5,0_7,1_0,1_7,1_8,1_9,1_{10},1_{11},1_{12},1_{13},1_{14}$
	75	20	22	10	1	$0_0,0_1,0_3,0_5,0_7,1_0,1_6,1_7,1_8,1_9,1_{10},1_{11},1_{12},1_{13},1_{14}$
	75	16	24	11	1	$0_0,0_1,0_3,0_5,0_7,1_0,1_5,1_6,1_7,1_8,1_9,1_{10},1_{11},1_{12},1_{13},3_7$
	75	12	28	13	3	$0_0,0_1,0_3,0_5,0_7,1_3,1_5,1_7,2_3,2_5,2_7,3_1,3_3,3_7,4_1,4_3,4_7$
	75	9	32	15	3	$0_1,0_3,0_5,0_7,1_0,1_3,1_5,1_7,2_3,2_5,2_7,3_1,3_3,3_7,4_1,4_3,4_7$

Notation : 'r_t' represents conjugacy class leader with index (r,t).

6.2 Decoding of codes of length 45

All codes discussed have area 3 x 15. The zeros of the code are given in terms of the leaders of the zero conjugacy classes. The decoding computation are done in $GF(2^4)$. This is an advantage over decoding of 1–D cyclic codes of length 45 which require $GF(2^{12})$ for decoding computation. We consider all the good codes of 45 listed in [56] in the sense of having maximum possible d_{min} among linear codes having same n and k. For some codes in list of [56] having type–3 decoding algorithm we have constructed another code with the same parameters having type–1 decoding algorithm. In such cases proofs of d_{min} are given based on 2–D complexity theorem.

6.2.1 Decoding of $(45, 25, 8)$

This code has $e_{2-D} = 3$ and decoding type is 1. The zero locations of the code are given by (1,0), (0,3), (0,5), (0,7), (1,7) and (2,7)} and their conjugates. The configuration of zeros is as shown in Figure 6.2.1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
1	Z							Z				Z		Z	Z
2	Z							Z				Z		Z	Z

Figure 6.2.1 Configuration of zeros of (45, 25, 8)

Derivation of decoding algorithm : The 6 consecutive zeros in row–0 can determine $\Lambda(z)$ if the column–wise error configuration is (o, o, o). If there is a even–weight column error configuration $\binom{o}{o}$, then 3 consecutive syndromes of row–1 can update $\Lambda(z)$ in the sense already explained. The recursive extension of syndromes is straightforward.

Errors–Correcting Algorithm for (45, 25, 8)

1) Pass the syndrome $S_{0,9}$ to $S_{0,14}$ through B–M algorithm and compute the row connection polynomial $\Lambda(z)$. Determine syndromes of row–0 by recursion.

2) If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{1,13}$ to $S_{1,0}$ through B–M algorithm. Use $\Lambda(z)$ to determine all the syndromes of rows 1 and 2 by recursion.

3) 2–D IDFT of S gives e.

Example 6.3.1: Consider an error pattern $e(x,y) = y^6 + xy^9 + x^2y^{12}$. $GF(2^4)$ is generated by (x^4+x+1) and α is a primitive 15th root of unity. The available syndromes are shown below in bold and italicized entries.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	α^{14}	α^{13}	α^{7}	α^{11}	1	α^{14}	α^{13}	α^{7}	α^{11}	1	α^{14}	α^{13}	α^{7}	α^{11}
1	0	α^{11}	α^{14}	α^{11}	α^{14}	0	α^{11}	α^{14}	α^{11}	α^{14}	0	α^{11}	α^{14}	α^{11}	α^{14}
2	0	α^{7}	α^{7}	α^{13}	α^{13}	0	α^{7}	α^{7}	α^{13}	α^{13}	0	α^{7}	α^{7}	α^{13}	α^{13}

Row–0 syndromes $S_{0,9}$ to $S_{0,14}$ produce $\Lambda(z) = (1 + \alpha^{14}z + \alpha^{8}z^{2} + \alpha^{12}z^{3})$. The recursive extension is straightforward and syndromes found from recursive extension are represented by non–bold and non–italicized entries.

Erasure Decoding of $(45, 25, 8)$

We have to use to entirely different approach to decoding if we wish to perform errors–and–erasures decoding upto full capability. The decoding algorithm for errors–and–erasures correction becomes type–3 from type–1 for errors–only correction. The first stage decoding is based on $(15, 11, 3)$ code. The different combination of errors–and–erasures corrected are given by $2e+r \leq 7$ for $e = 0$ to 3.

Step 1 tries to perform row decoding if a row has upto 2 erasures or 1 error. If e_k and r_k represents error and erasure for row–k, then $2e_k + r_k \leq 2$ for at least one $k = 0$ to 2. Therefore, at least, one row is properly decoded if $2e+r \leq 7$. Decoded row–j is assigned w_j equal to number of errors corrected; in case only erasures are corrected $w_j = 0$.

Step 2 forms $\phi(z)$ on the basis of two least reliable rows and finds $S_{0,0}$. Step 3 tries further correction of any rows which have 3 erasures or one row–erasure and one row–error. This is justified as $2e_k + t_k \leq 3$, for at least one of 2 rows with errors if $2e+r \leq 7$. Then 1–D DFT of columns of R' is taken to obtain S. If not every syndrome is 0 then $\phi(z)$ is found from $S_{0,1}$ and $S_{0,2}$ and the remaining syndromes are found.

Errors–and–Erasures Correcting Algorithm for $(45, 25, 8)$

1) Obtain R' by taking 1–D DFT of rows of r. For rows $k = 0$ to 2, process $R'_{k,7}$, $R'_{k,11}$, $R'_{k,13}$ and $R'_{k,14}$ and perform correction for 2 erasures or 1 error. If a row–k is properly decoded assign w_k equals the number of errors corrected; if k^{th} row is uncorrectable $w_k = 3$.

2)   Compute row–parity check vector p. Take 1–D DFT of p to get $S_{0,1}$ and $S_{0,2}$. Compute $\phi(z)$ from 2 rows having lowest reliability. Compute $S_{0,0}$. Obtain $R'_{k,0}$, for k=0 to 2.

3)   Using $R'_{k,0}$, for k=0 to 2, correct rows having 3 erasures or single error–and–erasure and identify uncorrectable rows.

4)   Compute 1–D DFT of the columns of R'. If no row is identified as uncorrectable, find $\phi(z)$ from $S_{1,0}$ and $S_{2,0}$. Use $\phi(z)$ by column recursive extension to find the remaining syndromes.

Example 6.3.2: Suppose all 0 word is transmitted and errors–and–erasures pattern is as shown below. Erasures are represented by x.

```
┌─────────────────────────────┐
│ 0 0 0 0 0 x 0 0 0 0 0 x x │
│ 0 0 0 0 1 0 0 0 x 0 0 0 0 0 0 │
│ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 │
└─────────────────────────────┘
```

The R' is given by

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	α^3	α^6	α^{13}	α^{12}	0	α^{11}	α^3	α^9	α^{14}	0	α^9	α^7	α^{12}	α^6
1	0	α^5	α^{10}	α^8	α^5	1	α	α^4	α^{10}	α^4	1	α^2	α^2	α	α^8
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The bold and italicized entries represent individual row syndromes available. Since row–0 has 3 erasures, it cannot be corrected in step 1. Similarly row–1 syndromes does not satisfy erasure polynomial $(1 + \alpha^8 z)$. row–2 is detected as having no errors. Hence $\phi(z) = (1 + \alpha^{10}z + \alpha^5 z^2)$. Row–parity check vector is $(1, 0, 0)$ and by taking its 1–D DFT we have $S_{0,1} = 1$ and $S_{0,2} = 1$. $S_{0,0}$ is computed to be 1. Hence $(1, 0, 0)$ are additional syndromes are available for all rows at the end of step 2. In step 3, it is found that row–0 syndromes satisfy erasure polynomial $(1 + \alpha^3 z + \alpha^9 z^2 + \alpha^3 z^3)$ and row–1 syndromes satisfy $(1 + \alpha^5 z + \alpha^{12} z^2)$. Step 4 is not required as all errors and erasures are corrected in the step 3 itself.

6.2.2 Decoding of $(45, 17-2p, 12+2p)$

We discuss 2 codes corresponding to p = 0 and 1. The $e_{2-D} = 5+p$ and 2–D decoding is of type–1.

The zero conjugacy class leaders for (45, 17, 12) are given by (0,1), (0,3), (0,5), (0,7), (1,1), (1,2), (1,3) and (1,5). The code (45, 15, 14) has an additional zero conjugacy class with leader (1,0). The configuration of zero locations is shown in Figure 6.2.2.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	(Z)	Z	Z	Z	Z	Z			Z		Z	Z	Z	Z	Z
2	(Z)	Z	Z		Z		Z		Z	Z	Z		Z		

Figure 6.2.2 Configurations of zeros of $(45, 17, 12)$

and $(45, 15, 14)$ [additional zeros enclosed in ( )]

Decoding Algorithm for $(45, 17{-}2p, 12{+}2p)$ $(p = 0$ or $1)$

1) Pass the syndromes $S_{0,1}$ to $S_{0,14}$ through B–M algorithm and compute $\Lambda(z)$. Determine $S_{0,0}$ using $\Lambda(z)$ by row recursion.

2) If $\deg(\Lambda(z)) \leq 3{+}p$, then update $\Lambda(z)$ by passing $S_{1,1{-}p}$ to $S_{1,5}$ through B–M algorithm. Determine syndromes of row 1 using $\Lambda(z)$ by row recursion. Then syndromes of row 2 are determined using conjugacy property and/or recursion.

3) 2–D IDFT of S gives e.

6.2.3 Decoding of $(45, 10, 16)$–I

We discuss $(45, 10, 16)$–I having type–3 decoding algorithm and $(45, 10, 16)$–II having type–1 decoding algorithm. Both are good codes having $e_{2\text{–}D} = 7$. We first discuss $(45, 10, 16)$–I which has 2–D decoding algorithm of type–3. The zeros are given by $(0,0)$, $(0,1)$, $(0,3)$, $(0,7)$, $(1,0)$, $(1,3)$, $(2,3)$, $(1,5)$, $(2,5)$, $(1,7)$ and $(2,7)$. The configuration of zeros is shown in Figure 6.2.3.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z		Z	Z	Z	Z		Z	Z	Z	Z
1	Z			Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
2	Z			Z		Z	Z	Z		Z	Z	Z	Z	Z	Z

Figure 6.2.3 Configuration of zeros of $(45, 10, 16)$–I

Derivation of decoding algorithm : The first stage consists of row decoding based on $(15, 6, 6)$ code which corrects upto 2 errors and detects 3 or more errors. A row error of weight ≥ 4 is either detected as uncorrectable or incorrectly decoded. A row error weight 6 or 7, in addition may go undetected. Each decoded row is assigned w_j and $\phi_{uc}(z)$ is formed from locators of uncorrectable rows.

From 1–D DFT of the columns of {T} to obtain S. The row–wise configuration of errors not corrected in the first stage are: $\left(\begin{smallmatrix} 8,8 \\ 8 \end{smallmatrix}\right)$ or $\left(\begin{smallmatrix} 8,8 \\ 8 \end{smallmatrix}\right)$ or a row error having 5 or more errors. If $\phi_{uc}(z) = 1$, then $\phi(z)$ is determined from $S_{1,5}$ and $S_{2,5}$, for if there are 2 incorrectly decoded rows, then the number of errors ≥ 8. If a single row is detected as having uncorrectable errors, then there can be at most one row incorrectly decoded. In this case $\phi(z)$ is constructed as the product of row locators of the uncorrectable row and least reliable decoded row. Finally if there two uncorrectable rows $\phi(z)$ is immediately known.

After $\phi(z)$ is determined $S_{0,5}$ and $S_{0,10}$ are computed. If the $\deg(\phi(z)) = 1$, then computing rest of the syndromes is straightforward. For the case of two uncorrectable rows, the B–M algorithm on the syndromes of row–0 followed by B–M algorithms on the syndromes of row–1 determines the $\Lambda(z)$, from which the rest of the syndromes are computed using $\Lambda(z)$. For the case of a single uncorrectable row and degree of $\phi(z) = 2$, more computation is necessary. This is because column spread of the errors may be 9 for some cases of error and there are not enough syndromes to handle this case. The procedure is to remove the errors introduced for incorrectly decoded row in step 1. Then computation of $\Lambda(z)$ is straightforward and the remaining syndromes are computed.

Decoding Algorithm for (45, 10, 16)–I

1) Obtain R' by taking 1–D DFT of rows of r. For rows i = 0 to 2, Process syndromes $R'_{i,j}$, for j = 11 to 0, using B–M algorithm. If for j^{th} row $\deg(\Lambda_j(z)) \leq 2$, then perform correction and assign w_j; otherwise detect that 3 or more errors have happened. Form $\phi_{uc}(z)$ from the locators the rows having uncorrectable errors.

2) Form 1–D DFT of columns of R' to obtain S.

3)   a)   If $\deg(\phi_{uc}(z)) = 0$, pass $S_{1,5}$ and $S_{2,5}$ through B–M algorithm to find $\phi(z)$. Determine the remaining syndromes using $\phi(z)$. Go to step 6.

       b)   If $\deg(\phi_{uc}(z)) = 1$, then $\phi(z) = \phi_{uc}(z)$ if $S_{1,5}$ and $S_{1,10}$ satisfy $\phi_{uc}(z)$. Complete recursion as above; else, form $\phi(z)$ as the product of $\phi_{uc}(z)$ and $\phi_1(z)$, where the root of $\phi_1(z)$ is the locator of least reliable decoded row, and go to step 4.

       c)   If $\deg(\phi_{uc}(z)) = 2$, the set $\phi(z) = \phi_{uc}(z)$. Go to step 5.

4) If $\deg(\phi_{uc}(z)) = 1$ and $\deg(\phi(z)) = 2$, then remove the errors in the incorrectly decoded row introduced during stage–1 decoding. Adapt the syndromes.

5) Determine $S_{0,5}$ and $S_{0,10}$ using $\phi(z)$. Pass $S_{0,1}$ to $S_{0,14}$ through B–M algorithm and find $\Lambda(z)$. Update $\Lambda(z)$ by passing $S_{1,9}$ to $S_{1,0}$ through B–M algorithm. Then compute the remaining syndromes using $\Lambda(z)$ by row recursion.

6) 2–D IDFT of S gives e.

6.2.4 Decoding of (45, 10, 16)–II

It is possible to change zero location of (45, 10, 16)–I from (1,5) or (2,5) to (0,5) and the resulting code can correct 7 errors and still have the same dimension. The zero locations are shown in Fig. 6.3.4. Moreover, as we now have consecutive 15 syndromes in row–0, 2–D decoding algorithm becomes type–1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z			Z		Z	Z	Z		Z		Z	Z	Z	Z
2	Z			Z			Z	Z		Z	Z	Z	Z	Z	Z

Figure 6.2.4 Configuration of zeros of (45, 10, 16)–II

To show that $d_{min} \geq 16$ proceed as follows. Since the first row of 2–D DFT of any codeword is all zero vector, the nonzero columns of the codeword must have weight at least 2. If the codeword has 7 nonzero columns, then by 2–D complexity theorem the codeword is seen to be zero. Therefore $d_{min} \geq 16$ as this is an even weight code.

Decoding Algorithm for (45, 10, 16)–II

1) Pass $\{S_{0,j}\}_{j=0}^{15}$ through B–M algorithm and find $\Lambda(z)$.

2) Update $\Lambda(z)$ by passing $\{S_{2,j}\}_{j=9}^{0}$ through B–M algorithm. Compute the syndromes of row 2 by row recursion using $\Lambda(z)$. Compute the syndromes of row 1 using conjugacy property.

3) 2–D IDFT of S gives e.

6.2.5 Decoding of (45, 8, 20)

This code has zero conjugacy class leaders (0,0), (0,1), (0,3), (0,5), (0,7), (1,1), (1,3), (1,5), (1,7), (2,1) and (2,3). The configuration of zero locations is as shown in Figure 6.2.5. The decoding is of type–1.

Decoding Algorithm for (45, 8, 20)

1) Compute the column parity checks. Find $\Lambda(z)$ corresponding to locators of columns having odd parity–checks.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1		Z	Z	Z	Z	Z	Z	Z	Z	Z			Z	Z	Z
2		Z	Z	Z			Z		Z	Z	Z	Z	Z		Z

Figure 6.2.5 Configuration of zeros of (45, 8, 20)

2)   If $\deg(\Lambda(z)) \leq 7$, then update $\Lambda(z)$ by passing $S_{1,1}$ to $S_{1,9}$ through B–M algorithm. Complete the recursion for row 1 using $\Lambda(z)$ and use conjugacy property to compute the syndromes of row 2.

3)   2–D IDFT of S gives e.

6.2.6 Decoding of (45, 6, 22)–I

For the (45, 6, 22) we consider two different codes having different zero locations. (45, 6, 22)–II and (45, 8, 20) code have type–1 decoding algorithm. The decoding of (45, 6, 22)–I is type–3. The zero conjugacy class leaders for (45, 6, 22)–I are given by (0,0), (0,3), (0,5), (0,7), (1,1), (1,3), (1,5), (1,7),(2,1), (2,3), (2,5) and (2,7). The configuration of zeros is as shown in Figure 6.2.6.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z			Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
1		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
2		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Figure 6.2.6 Configuration of zeros of (45, 6, 22)–I

Derivation of decoding algorithm : The first stage decoding is based on (15, 5, 7) code which corrects upto 3 errors. A row error of weight 4, 5 or 6 is either incorrectly decoded or detected as uncorrectable. A row error of weight 7, 8, 9 or 10, in addition to above alterations, may also go undetected. Decoded row–j is assigned w_j equal to the number of corrections made. The $\phi_{uc}(z)$ is formed from locators of rows detected as uncorrectable.

In the second stage decoding the situation is analyzed as follows. If all 3 rows output a correctable pattern then there can be at most two incorrectly decoded or uncorrectable rows. Then $\phi(z)$ is formed from locators of two least reliable decoded rows. If one row is detected as uncorrectable, then also the incorrectly decoded row identified based on highest correction number. In case of 2 uncorrectable rows the $\phi(z)$ computation is straightforward. From $\phi(z)$ the remaining syndromes of row–0 at positions 1, 2, 4 and 8 are computed.

Finally, if a row is known to be incorrectly decoded, errors introduced are removed and syndromes are adapted. $\Lambda(z)$ is compute in a straightforward manner and $S_{1,0}$ and $S_{2,0}$ are computed.

Decoding Algorithm for (45, 6, 22)–I

1) Obtain R' by taking 1–D DFT of rows of r. For rows $i = 0$ to 2, process $\{R'_{i,j}\}_{j=9}^{14}$ using B–M algorithm and correct row–i if $\deg(\Lambda_i(z)) \leq 3$. Assign w_i for decoded i^{th} row. Form $\phi_{uc}(z)$ from the locators of the uncorrectable rows.

2) Form the 1–D DFT of columns of R' to obtain S.

3) a) If $\deg(\phi_{uc}(z)) = 1$, then $\phi(z) = \phi_{uc}(z)$ if $S_{1,1}$ and $S_{2,1}$ satisfy $\phi_{uc}(z)$. Determine the rest of the syndromes using $\phi(z)$; else, form $\phi(z)$ as the product of $\phi_{uc}(z)\phi_1(z)$ where the root of $\phi_1(z)$ is the locator of least reliable decoded row. Compute $S_{0,1}$, $S_{0,2}$, $S_{0,4}$ and $S_{0,8}$ and go to step 4.

   b) If $\deg(\phi_{uc}(z)) = 2$, the set $\phi(z) = \phi_{uc}(z)$. Compute $S_{0,1}$, $S_{0,2}$, $S_{0,4}$ and $S_{0,8}$ and go to step 5.

   c) If $\deg(\phi_{uc}(z)) = 0$, form $\phi(z)$ from locators of two least reliable corrected rows. Find $S_{0,1}$, $S_{0,2}$, $S_{0,4}$ and $S_{0,8}$. Go to step 4.

4) For rows known to be incorrectly decoded remove the errors introduced during stage–1 decoding. Adapt the syndromes.

5) Take 1–D IDFT of row–0 of S and form $\Lambda(z)$ from locators of nonzero positions. If $\deg(\Lambda(z)) \leq 8$, then $\Lambda(z)$ is updated by passing $S_{1,1}$ to $S_{1,14}$ through B–M algorithm. $S_{1,0}$ and $S_{2,0}$ are computed using $\Lambda(z)$.

6) 2–D IDFT of S gives e.

Example 6.2.3: Suppose the transmitted code vector is / 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 / 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 / 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0. Let $e(x,y) = (1+y+y^3+y^6) + x(y^2+y^5+y^7+y^{14}) + x^2(y^4+y^9)$. The R' syndromes that are processed in the first stage of individual row decoding is as shown below.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				α^7		α^{10}	α^{14}	α^{14}		α^{11}	α^5	α^7	α^{13}	α^{11}	α^{13}
1				α^{11}		1	α^7	α^{14}		α^{13}	1	α^7	α^{14}	α^{11}	α^{13}
2				0		α^{10}	0	α^8		0	α^5	α^4	0	α^2	α

First stage (15, 5, 7) decoding for row–0 syndromes produces an uncorrectable error pattern; row–1 syndromes satisfy $\Lambda_1(z) = (1 + \alpha^7 z + \alpha^4 z^2 + \alpha^6 z^3)$ and $w_1 = 3$; row–2 syndromes satisfy $(1 + \alpha^{14} z + \alpha^{13} z^2)$ and $w_2 = 2$. Therefore $\phi_{uc}(z) = (1 + z)$. Taking 1–D DFT of columns of R^t we obtain $S_{1,1} = \alpha^3$ and $S_{2,1} = \alpha^4$ which does not satisfy $\phi_{uc}(z)$. From step 3a) we find $\phi(z) = \phi_{uc}(z)(1 + \alpha^5 z) = (1 + \alpha^{10} z + \alpha^5 z^2)$. $S_{0,1}$, $S_{0,2}$, $S_{0,4}$ and $S_{0,8}$ are found using $\phi(z)$. Since row–1 is known to be incorrectly decoded, errors introduced earlier in step 1 is removed and syndromes are adapted accordingly. Finally at the end of step 4, the following syndromes in bold and italicized entries are known.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	α^{10}	α^5	α^8	α^{10}	α^5	α	0	α^5	α^4	α^{10}	0	α^2	0	0
1	0	α^{10}	α^5	α^{14}	α^{10}	1	α^5	α^9	α^5	α^5	0	α^2	α^{11}	α^6	α^8
2	0	α^{10}	α^5	α^{10}	α^{10}	0	α^{13}	α^4	α^5	α^7	1	α^{12}	α^{10}	α	α^3

Step 5 finds $\Lambda(z) = (1 + \alpha^{10} z + \alpha^{10} z^2 + \alpha z^3 + \alpha^9 z^4 + \alpha^{10} z^5 + \alpha^7 z^6 + \alpha^8 z^8)$. $S_{1,0}$ and $S_{2,0}$ are found to be 0 using $\Lambda(z)$ as shown by entries which are non–bold. Once S is known computation of c is straightforward.

6.2.7 Decoding of (45, 6, 22)–II

An alternative (45, 6, 22)–II code is possible which leads to simple type–1 decoding algorithm. The zero location given by (0,0), (0,1), (0,3), (0,5), (0,7), (1,3), (1,5), (1,7),(2,1), (2,3), (2,5) and (2,7). The configuration of zero locations is as shown in Figure 6.2.7.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1			Z	Z		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
2		Z		Z	Z	Z	Z	Z		Z	Z	Z	Z	Z	Z

Figure 6.2.7 Configurations of zeros of (45, 6, 22)–II

To show $d_{min} \geq 22$ proceed as follows. As first row is all zeros the nonzero column must have weight at least 2. Now if a codeword has ≤ 10 columns, then by 2–D complexity theorem the codeword is seen to be zero. Therefore $d_{min} \geq 22$ as this is an even weight code.

Decoding Algorithm for (45, 6, 22)–II

1) Compute column parity checks. Form $\Lambda(z)$ having as roots the locators of columns with odd parity–check.

2)   If $\deg(\Lambda(z)) \leq 8$, then update $\Lambda(z)$ by passing $S_{1,5}$ to $S_{1,14}$ through B–M algorithm. Complete the recursion for row 1 and compute syndromes of row 2 using conjugacy property.

3)   2–D IDFT of S gives e.

6.3 Decoding of codes of length 49

The area of the code is 7 x 7. The decoding computation are done in $GF(2^3)$. This is an advantage over decoding of 1–D cyclic codes of length 49 which require $GF(2^{21})$ for decoding computation. We consider almost all the good codes listed in [56]. For cases where decoding is of type–1, it is necessary to use both $\Lambda(z)$ and $\phi(z)$ for recursive extension unlike type–1 decoding in 45 length codes. In some cases though connection polynomial is easily found recursive extension presents a problem. So we have to first compute syndromes of rows and columns wherever possible and apply 1–D decoding techniques to correct errors. This type of 2–D algorithms can be considered as a version of type–3.

6.3.1 Decoding of $(49, 34, 6)$

This is code has $e_{2-D} = 2$ and 2–D decoding algorithm is of type–1. The conjugacy leaders for zero locations are given by (0,1), (0,3), (1,0), (3,0), and (3,6). The configuration of zero locations is as shown in Figure 6.3.1.

r/c	0	1	2	3	4	5	6
0		Z	Z	Z	Z	Z	Z
1	Z						
2	Z						
3	Z						Z
4	Z						
5	Z			Z			
6	Z					Z	

Figure 6.3.1 Configuration of zeros of (49, 34, 6)

Decoding Algorithm for (49, 34, 6)

1)   Pass $S_{0,0}$ to $S_{0,6}$ through B–M algorithm and find $\Lambda(z)$. If $\deg(\Lambda(z)) = 0$, then determine $\Lambda(z)$ by passing $S_{3,6}$ to $S_{3,0}$ through B–M algorithm. Compute $S_{0,0}$ using $\Lambda(z)$.

2)   Determine the syndromes of row 3 by recursive computation. Use conjugacy property to determine syndromes of row 5 and row 6.

3) Pass $S_{0,0}$ to $S_{6,0}$ through B–M algorithm and find $\phi(z)$. If $\deg(\phi(z)) = 0$, then determine $\phi(z)$ by passing $S_{5,1}$ to $S_{0,1}$ through B–M algorithm.

4) Determine the remaining syndromes by column recursion using $\phi(z)$.

5) 2–D IDFT of S gives e.

6.3.2 Decoding of $(49, 30, 8)$

This code has $e_{2-D} = 3$. 2–D decoding algorithm is of type–1. The conjugacy class leaders of zero locations are given by (0,0), (0,1), (0,3), (1,0), (3,0), (3,5) and (3,6). The configuration of zeros is as shown in Figure 6.3.2.

r/c	0	1	2	3	4	5	6
0	Z	Z	Z	Z	Z	Z	Z
1	Z						
2	Z						
3	Z					Z	Z
4	Z						
5	Z		Z				Z
6	Z		Z		Z		

Figure 6.3.2 Configuration of zeros of $(49, 30, 8)$

Decoding Algorithm for (49, 30, 8)

1) Pass $S_{0,0}$ to $S_{0,6}$ through B–M algorithm and find $\Lambda(z)$. If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{3,5}$ to $S_{3,0}$ through B–M algorithm.

2) Determine the syndromes of row 3 by recursive computation. Use conjugacy properties to determine syndromes of row 5 and row 6.

3) Pass $S_{0,0}$ to $S_{6,0}$ through B–M algorithm and find $\phi(z)$. If $\deg(\phi(z)) \leq 1$, then update $\phi(z)$ by passing $S_{5,3}$ to $S_{0,3}$ through B–M algorithm.

4) Determine the remaining syndromes by column recursion using $\phi(z)$.

5) 2–D IDFT of the S gives e.

Example 6.3.1: Suppose the transmitted code vector is /0 1 0 1 0 0 0/ 1 0 1 0 0 0 0 / 0 0 0 0 0 1 1/ 0 0 1 0 1 0 0/ 1 1 1 0 1 0 0/ 1 0 1 1 0 0 1/ 1 0 0 0 0 1 0. Let α be 7th root of unity in $GF(2^3)$ generated by (x^3+x+1). Let $e(x,y) = (xy + x^2 y^3 + x^3 y^4)$. The syndromes are shown below in bold and italicized entries.

r/c	0	1	2	3	4	5	6
0	1	α^5	α^9	0	α^6	0	0
1	α^6	α	α^5	α	1	α	1
2	α^5	1	α^2	α^2	α^3	1	α^2
3	α	α^5	α^3	α^4	1	1	0
4	α^9	α^6	1	1	α^4	α^4	α^4
5	α^4	α^5	1	0	α^6	α^2	1
6	α^2	1	α^3	1	α^6	0	α

From row–0 syndromes $\Lambda(z)$ is found to be $(1 + \alpha^5 z + \alpha z^3)$. Using $\Lambda(z)$, syndromes for rows 3, 5 and 6 are determined. From column–1 syndromes $\phi(z)$ is found to be $(1 + \alpha^6 z + \alpha z^2 + \alpha^6 z^3)$. Since 3 consecutive syndromes are known for each column the remaining syndromes are found by using $\phi(z)$ by column recursion.

The code obtained by dropping $C_{0,0}$ position has parameters $(49, 31, 7)$ is also listed in [56]. The decoding algorithm for this code changes only at the first step where $\Lambda(z)$ is computed by passing $S_{0,1}$ to $S_{0,6}$ through B–M algorithm and that $S_{0,0}$ has to be computed from recursive extension.

6.3.3 Decoding of $(49, 21, 12)$

This is a 5 error correcting code. 2–D decoding algorithm for this case is a variation of type–3. The leaders of zero conjugacy class are given by $(0,0)$, $(0,1)$, $(0,3)$, $(1,0)$, $(1,6)$, $(3,0)$, $(3,3)$, $(3,4)$, $(3,5)$ and $(3,6)$. The configuration of zeros is as shown in FIgure 6.3.3.

r/c	0	1	2	3	4	5	6
0	Z	Z	Z	Z	Z	Z	Z
1	Z						Z
2	Z					Z	
3	Z			Z	Z	Z	Z
4	Z			Z			
5	Z		Z	Z		Z	Z
6	Z	Z		Z		Z	Z

Figure 6.3.3 Configuration of zeros of $(49, 21, 12)$

Derivation of decoding algorithm : Though the structure of zero locations make it simple to find the $\Lambda(z)$ or $\phi(z)$ it is not straightforward to determine all the syndromes using either row or column recursive

extension for some cases of error. We propose the following method to correct errors. First determine the syndromes of rows and columns, wherever possible. Then process 1–D IDFT of column (row) syndromes by 1–D decoding techniques to determine the row (column) error patterns.

For the present case, recursive extension is not possible if both the $\Lambda(z)$ and $\phi(z)$ are of degree ≥ 4. If $\deg(\Lambda(z)) = 4$, then at most one column can have an error of weight 2 if upto 5 errors have occurred. The decoding procedure is to take 1–D IDFT of columns 0, 3, 5 and 6. These give 3 consecutive syndromes for each row which either correct a single error or detect a double error. Adapt the syndromes if a row error is corrected. Find the rest of the syndromes by column recursion.

Decoding Algorithm for (49, 21, 12)

1) Compute the column parity checks and form $\Lambda(z)$ having as roots locators of columns with odd–parity checks. If $\deg(\Lambda(z)) \leq 3$, then update $\Lambda(z)$ by passing $S_{3,3}$ to $S_{3,0}$ through B–M algorithm.

2) Determine the syndromes of row 3 by recursive computation. Use conjugacy constraints to determine syndromes of row 5 and row 6.

3) Compute the row parity checks and form $\phi(z)$ having as roots the locators of rows with odd–parity checks. If $\deg(\phi(z)) \leq 3$, then update $\phi(z)$ by passing $S_{3,3}$ to $S_{0,3}$ (column 3) through B–M algorithm.

4) Determine the syndromes of column–3 by recursive extension. Use conjugacy constraints to determine syndromes of columns 5 and 6.

5) If $\deg(\Lambda(z)) \leq 3$ then determine rest of the syndromes using $\Lambda(z)$. If the $\deg(\phi(z)) \leq 3$, then determine rest of the syndromes using $\phi(z)$. Go to step 7.

6) Compute 1–D IDFT of rows 0, 3, 5 and 6. For columns having odd parity do single error correction. Adapt the syndromes. Find the row connection polynomial $\Lambda(z)$ which can be of degree at most 1. Determine the rest of the syndromes by recursive extension using $\Lambda(z)$.

7) 2–D IDFT of S gives e.

Example 6.3.2: Suppose the transmitted code vector is /0 0 0 1 0 1 0/ 0 0 1 0 0 1 0 / 0 1 1 0 1 0 1/ 1 1 0 1 1 0 0/ 0 1 1 0 0 0 0/ 0 0 0 0 1 0 1/ 1 1 1 0 1 0 0. Let α be 7th root of unity in $GF(2^3)$ generated by (x^3+x+1). Let $e(x,y) = (xy + x^2y^2 + x^3y^3 + (x^4+x^5)y^4)$. The available syndromes are shown below in bold and italicized entries.

r/c	0	1	2	3	4	5	6
0	1	α^6	α^5	α	α^3	α^4	α^2
1	α^2	1	0	α^2	α	1	α
2	α^4	α^2	1	1	0	α^2	α^4
3	α^5	0	α^5	α	α^3	α	α^3
4	α	0	α^4	α^4	1	α	1
5	α^6	α^6	α^5	α^5	0	α^4	α^4
6	α^3	α^6	0	α^2	α^3	α^6	α^2

From row–0 syndromes $\Lambda(z)$ is found to be $(1 + \alpha^6 z + \alpha z^2 + \alpha^6 z^3)$. Five consecutive syndromes of row–3 updates $\Lambda(z)$ to $(1 + \alpha^3 z + z^2 + \alpha z^3 + \alpha^3 z^4)$. Syndromes of rows 3, 5 and 6 are computed using $\Lambda(z)$. Computing column–parity vector p, we see that 1, 2 and 3 columns are in errors. Find 1–D IDFT of rows 0, 3, 5 and 6. For column positions 1, 2 and 3, 1–D IDFT coefficients turn out to be respectively, $(\alpha^5, \alpha^6, 1)$, $(\alpha^3, \alpha^5, 1)$ and $(\alpha, \alpha^4, 1)$ and consequently row positions of errors are known. Performing single error correction and adapting the syndromes it is found that $\Lambda(z) = (1 + \alpha^4 z)$ from which all the remaining syndromes can be determined.

6.3.4 Decoding of $(49, 15, 14)$

This code has $e_{2-D} = 6$. 2–D decoding algorithm is of type–3. The conjugacy class leaders of zero locations are given by $(0,0)$, $(1,0)$, $(1,4)$, $(1,5)$, $(1,6)$, $(3,0)$, $(3,1)$, $(3,2)$, $(3,3)$, $(3,4)$, $(3,5)$ and $(3,6)$. The configuration of zeros is as shown in Figure 6.3.4.

r/c	0	1	2	3	4	5	6
0	Z						
1	Z				Z	Z	Z
2	Z	Z		Z		Z	
3	Z	Z	Z	Z	Z	Z	Z
4	Z		Z	Z			Z
5	Z	Z	Z	Z	Z	Z	Z
6	Z	Z	Z	Z	Z	Z	Z

Figure 6.3.4 Configuration of zeros of $(49, 15, 14)$

Derivation of decoding algorithm : The first stage decoding consists of column decoding based on $(7, 4, 3)$ code which corrects single errors. If a correctable pattern is produced by the decoder, then assign w_j equal

to the number of errors corrected. A particular column may either be corrected or detected as having an uncorrectable error pattern or incorrectly decoded. Form $\Lambda_{uc}(z)$ from locators of column detected as uncorrectable.

The column–wise error configurations not corrected in the first stage are: $\left(\begin{smallmatrix} o & o & o \\ o & o & o \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 8 & o \\ 8 & o \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 8 & 8 \\ 8 & 8 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 8 & 8 \\ & 8 \end{smallmatrix}\right)$

and a single column error pattern having 5 or 6 errors. We have 4 consecutive syndromes in row–1 from which one can determine $\Lambda(z)$ if upto 2 columns are in error; otherwise $\Lambda(z)$ is formed from 3 least reliable corrected columns. Once $\Lambda(z)$ is known all the syndromes of row 1,2 and 4 are determined. To determine the syndromes of row 0 it is necessary to remove errors introduced by the column decoder in case of incorrect decoding and adapt the syndromes. Then $\phi(z)$ is then determined in a straightforward way.

Decoding Algorithm for (49, 15, 14)

1) Compute 1–D DFT of the columns of r to obtain $R'_{i,j}$ for columns j = 0 to 6, at locations i ϵ {3,6,5}. Process based on decoding of (7, 4, 3) code. The jth column, if corrected, assigned w_j equal to number of errors corrected. Form $\Lambda_{uc}(z)$ from locators of columns which are uncorrectable.

2) Find the 1–D DFT of rows of R' to obtain S.

3) Compute $\Lambda(z)$ by passing $S_{1,4}$ to $S_{1,0}$ through B–M algorithm with $\Lambda_{uc}(z)$ as the initialization polynomial. If $\Lambda(z)$ does not satisfy syndromes of row–1, then form $\Lambda(z)$ from 3 least reliable decoded columns.

4) Determine the syndromes of rows 1, 2 and 4 using $\Lambda(z)$ by row recursive extension.

5) If there are incorrectly decoded rows, then remove the errors added in step 1 and adapt the syndromes.

6) Compute 1–D IDFT of column–0 of S and form $\phi(z)$ from the locators of its nonzero positions. If $\deg(\phi(z)) \leq 4$, then update $\phi(z)$ by passing $S_{1,1}$ to $S_{6,1}$ through B–M algorithm.

7) Determine the syndromes of row 0 by column recursive extension using $\phi(z)$.

8) 2–D IDFT of S gives e.

6.3.5 Decoding of (49, 13, 16)

This code has $e_{2-D} = 7$. 2–D decoding algorithm is of type–3. The leaders of zero conjugacy class are given by (0,1), (0,3), (1,4), (1,5), (1,6),(3,0), (3,1), (3,2), (3,3), (3,4), (3,5) and (3,6).

r/c	0	1	2	3	4	5	6
0		Z	Z	Z	Z	Z	Z
1					Z	Z	Z
2		Z		Z		Z	
3	Z	Z	Z	Z	Z	Z	Z
4		Z	Z				Z
5	Z	Z	Z	Z	Z	Z	Z
6	Z	Z	Z	Z	Z	Z	Z

Figure 6.3.5 Configuration of zeros of (49, 13, 16)

The configuration of zero locations is as shown in Figure 6.3.5.

<u>Derivation of decoding algorithm</u> : The first stage consists of column decoding based on (7, 4, 3) code. The corrected columns are assigned correction numbers. Form $\Lambda_{uc}(z)$ based on locators of columns detected as uncorrectable.

In the second stage decoding at most 3 columns could be in error. The $\Lambda(z)$ is determined from 6 syndromes of row–0 and 3 consecutive syndromes of row–1 with $\Lambda_{uc}(z)$ as initialization polynomial. If $\Lambda(z)$ does not satisfy syndromes of row–0 and row–1, then $\Lambda(z)$ is formed from 3 least reliable decoded columns of first stage decoding. Note that for this code $\Lambda(z)$ suffices for determining rest of the syndromes. It is not necessary to remove errors as in the case of decoding algorithm for (49, 15, 14).

<div align="center">Decoding Algorithm for (49, 13, 16)</div>

1) Compute 1–D DFT of columns of r to obtain R'. For columns j = 0 to 6, process $R'_{3,j}$, $R'_{5,j}$ and $R'_{6,j}$ based on (7, 4, 3) code. The j$^{\text{th}}$ column, if corrected, assigned w_j equal to number of errors corrected. Form $\Lambda_{uc}(z)$ from locators of columns which are uncorrectable.

2) Find the 1–D DFT of rows of R' to obtain S.

3) Pass $S_{0,1}$ to $S_{0,6}$ through B–M algorithm to find $\Lambda(z)$ with $\Lambda_{uc}(z)$ as the initialization polynomial. If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{1,4}$ to $S_{1,6}$ through B–M algorithm. If $\Lambda(z)$ does not satisfy syndromes of row–0 and row–1, then form $\Lambda(z)$ from 3 least reliable columns.

4) Determine the remaining syndromes of rows 0, 1, 2 and 4 by row recursion using $\Lambda(z)$.

5) 2–D IDFT of S gives e.

6.3.6 Decoding of (49, 10, 20)

This code has $e_{2-D} = 9$. 2–D decoding algorithm is of type–3. The conjugacy leaders of zero locations are given by (0,1), (0,3), (1,0), (1,4), (1,5), (1,6),(3,0), (3,1), (3,2), (3,3), (3,4), (3,5) and (3,6).

r/c	0	1	2	3	4	5	6
0		Z	Z	Z	Z	Z	Z
1	Z				Z	Z	Z
2	Z	Z		Z		Z	
3	Z	Z	Z	Z	Z	Z	Z
4	Z		Z	Z			Z
5	Z	Z	Z	Z	Z	Z	Z
6	Z	Z	Z	Z	Z	Z	Z

Figure 6.3.6 Configuration of zeros of (49, 10, 20)

The configuration of zero locations is as shown in Figure 6.3.6.

Derivation of decoding algorithm : The first stage consists of column decoding based on (7, 4, 3) code. The corrected column are assigned correction numbers and the $\Lambda_{uc}(z)$ is formed from locators of uncorrectable columns.

In second stage upto 4 columns could be in error. First $\Lambda(z)$ is determined from 6 syndromes of row—0 and if necessary, updated using 4 consecutive syndromes of row—1, starting with $\Lambda_{uc}(z)$ as initialization polynomial. If $\Lambda(z)$ is found to be not valid, then 4 least reliable columns are used for recursive extension. That is enough to determine all the syndromes.

Decoding Algorithm for (49, 10, 20)

1) Compute 1—D DFT of columns of r to obtain $R'_{i,j}$ for $j = 0$ to 6, at locations $i \in \{3, 6, 5\}$. Process these syndromes based on (7, 4, 3) code. The jth column, if corrected, assigned w_j equal to the number of errors corrected. Form $\Lambda_{uc}(z)$ from locators of columns which are uncorrectable.

2) Find the 1—D DFT of rows of R' to obtain S.

3) Pass $S_{0,1}$ to $S_{0,6}$ through B—M algorithm to find $\Lambda(z)$ with $\Lambda_{uc}(z)$ as the initialization polynomial. If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{1,4}$ to $S_{1,0}$ through B—M algorithm. If $\Lambda(z)$ does not satisfy syndromes of row—0 and row—1, then form $\Lambda(z)$ from 4 least reliable columns.

4) Determine remaining syndromes of rows 0, 1, 2, and 4 using $\Lambda(z)$.

5) 2—D IDFT of S gives e.

6.3.7 Decoding of (49, 6, 24)

This code has $e_{2-D} = 11$. The 2—D decoding is of type 3. The conjugacy class leaders of zero locations are given by (0,0), (0,1), (0,3), (1,0), (1,3), (1,4), (1,5), (1,6),(3,0), (3,1), (3,2), (3,3), (3,4), (3,5)

r/c	0	1	2	3	4	5	6
0	Z	Z	Z	Z	Z	Z	Z
1	Z			Z	Z	Z	Z
2	Z	Z		Z		Z	Z
3	Z	Z	Z	Z	Z	Z	Z
4	Z		Z	Z		Z	Z
5	Z	Z	Z	Z	Z	Z	Z
6	Z	Z	Z	Z	Z	Z	Z

Figure 6.3.7 Configuration of zeros of (49, 6, 24)

and (3,6). The configuration of zero locations is as shown in 6.3.7.

Derivation of decoding algorithm: The first stage consists of column decoding based on (7, 3, 4) code. The corrected columns are assigned correction numbers and the $\Lambda_{uc}(z)$ is found from locators of uncorrectable columns.

In second stage at most 5 columns could be in error if upto 11 errors occur. First $\Lambda(z)$ is determined using 6 syndromes of row–0 with $\Lambda_{uc}(z)$ as the initialization polynomial and, if necessary, 5 consecutive syndromes of row–1. If $\Lambda(z)$ is not valid, then $\Lambda(z)$ is formed from 5 least reliable decoded columns and the remaining syndromes are found using $\Lambda(z)$.

<div align="center">Decoding Algorithm for (49, 6, 24)</div>

1) Find 1–D DFT of columns of r to obtain $R'_{i,j}$ for columns $j = 0$ to 6, at locations $i \in \{0, 3, 6, 5\}$. Process these syndromes based on (7, 3, 4) code. The jth column, if corrected, assigned w_j equal to the number of errors corrected. Form $\Lambda_{uc}(z)$ from the locators of columns which are uncorrectable.

2) Find the 1–D DFT of rows of R' to obtain S.

3) If $\deg(\Lambda_{uc}(z)) \leq 3$, pass $S_{0,0}$ to $S_{0,6}$ through B–M algorithm to find $\Lambda(z)$ with $\Lambda_{uc}(z)$ as the initialization polynomial. If $\deg(\Lambda(z)) \leq 3$, then update $\Lambda(z)$ using $S_{1,3}$ to $S_{1,0}$. If $\Lambda(z)$ found to be invalid, then form $\Lambda(z)$ from the locators of 5 least reliable decoded columns.

4) Determine the remaining syndromes of rows 1, 2 and 4 using $\Lambda(z)$.

5) 2–D IDFT of S gives e.

6.4 Decoding of codes of length 75

All the codes are assumed to be of area 5 x 15. The decoding computation are done in $GF(2^4)$. This has to be contrasted to codes of length 75 with area 3 x 25 which require decoding computation to be done in $GF(2^{20})$. Most of the codes we consider from [56] have type–1 decoding algorithm and some of these

codes have best possible minimum distance among comparable linear codes. We also consider some codes having type–3 decoding algorithm.

6.4.1 Decoding of $(75, 58, 6)$

This code has $e_{2-D} = 2$ and 2–D decoding algorithm is of type–1. The conjugacy class leaders of the zero locations are given by $(0,0)$, $(0,3)$, $(0,7)$, $(1,0)$ and $(1,14)$. The configuration of zero locations is as shown in Figure 6.4.1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z			Z			Z	Z		Z		Z	Z	Z	Z
1	Z														Z
2	Z													Z	
3	Z							Z							
4	Z											Z			

Figure 6.4.1 Configuration of zeros of $(75, 58, 6)$

Decoding Algorithm for $(75, 58, 6)$

1) Pass $S_{1,11}$ to $S_{1,14}$ through B–M algorithm and find $\Lambda(z)$. Determine the rest of syndromes of row–0 by recursion using $\Lambda(z)$.

2) If $\Lambda(z) = 1$, then determine $\Lambda(z)$ from $S_{1,14}$ and $S_{1,0}$.

3) Compute the syndromes of row 1 by recursive extension using $\Lambda(z)$. Then determine the syndromes of the remaining rows using conjugacy property and/or recursive extension.

4) 2–D IDFT S gives e.

6.4.2 Decoding of $(75, 52, 8)$

This code has $e_{2-D} = 3$ errors and 2–D decoding is of type–1. The conjugacy class leaders of the zero locations are given by $(0,0)$, $(0,3)$, $(0,5)$, $(0,7)$, $(1,0)$, $(1,13)$ and $(1,14)$. The configuration of zero positions is as shown in Figure 6.4.2.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z			Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
														Z	Z
1	Z											Z		Z	
2	Z														Z
3	Z							Z							
4	Z							Z				Z			

Figure 6.4.2 Configuration of zeros of $(75, 52, 8)$

Decoding Algorithm for (75, 52, 8)

1) Pass $S_{1,9}$ to $S_{1,14}$ through B–M algorithm and find $\Lambda(z)$. Determine the rest of syndromes of row–0 by recursion.

2) If $\deg(\Lambda(z)) \leq 1$, then update $\Lambda(z)$ by passing $S_{1,13}$ to $S_{1,0}$ through B–M algorithm.

3) Compute the syndromes of row 1 by recursive extension using $\Lambda(z)$. Then determine the syndromes of the remaining rows by conjugacy property.

4) 2–D IDFT of S gives e.

6.4.3 Decoding of (75, 46, 10)

This code has $e_{2-D} = 4$ and 2–D decoding is of type–1. The conjugacy class leaders of the zero locations are given by (0,0), (0,1), (0,3), (0,7), (1,0), (1,12), (1,13) and (1,14). The configuration of zero locations is as shown in Figure 6.4.3.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z		Z	Z	Z	Z		Z	Z	Z	Z
1	Z												Z	Z	Z
2	Z									Z		Z		Z	
3	Z						Z	Z							Z
4	Z			Z				Z				Z			

Figure 6.4.3 Configuration of zeros of (75, 46, 10)

Decoding Algorithm for (75, 46, 10)

1) Pass $S_{1,11}$ to $S_{1,4}$ through B–M algorithm and find $\Lambda(z)$. Determine the rest of syndromes of row–0 by recursion.

2) If $\deg(\Lambda(z)) \leq 2$, then update $\Lambda(z)$ by passing $S_{1,12}$ to $S_{1,0}$ through B–M algorithm.

3) Compute the syndromes of row 1 by recursive extension using $\Lambda(z)$. Then determine the syndromes of the remaining rows by conjugacy property.

4) 2–D IDFT of S gives e.

Example 6.4.1: Consider that transmitted word is given by /0 0 1 1 1 1 1 1 1 1 0 1 1 1 1/ 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0/ 1 1 1 0 0 0 0 1 0 0 1 1 1 1/ 1 0 1 0 0 1 0 1 1 0 1 1 0 1 0/ 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1. Let α represent 15th root of unity in GF(2^4) generated by (x^4+x+1). For $e(x,y) = (xy^8 + x^2y^3 + x^3 + x^4y^{11})$ and the syndromes are shown below in bold and italicized entries.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	α	α^2	α^{14}	α^4	0	α^{13}	α^{10}	α^8	α^7	0	α^5	α^{11}	α^{10}	α^5
1	1	α^7	α^8	α^8	α^9	0	α^2	α^8	α^{10}	α^{12}	α^{10}	α^{12}	α^8	α^{14}	α^{10}
2	1	α^5	α^{14}	α^9	α	α^5	α	α^9	α^3	α	0	α^{13}	α^4	α^5	α
3	1	α^4	α^{12}	α	α^5	α^5	α^4	α^5	α^{11}	α^4	0	α^4	α^6	α^6	α^7
4	1	α^6	α^{10}	α^2	α^{13}	0	α^3	α^{11}	α^2	α^8	α^{10}	α^{10}	α^2	α^2	α^3

9 consecutive syndromes of row–0 when passed through B–M algorithm produce $\Lambda(z) = (1 + \alpha z + \alpha^{10}z^2 + \alpha^{12}z^3 + \alpha^7 z^4)$. Since row–1 has 4 consecutive syndromes recursive extension is direct. Then we use conjugacy property to either determine enough syndromes so that recursion can be applied or determine all the syndromes using conjugate property itself. The syndromes found in this way are shown by entries which are non–bold and non–italicized type. Once S is known computation of c is direct.

6.4.4 Decoding of codes $(75, 41\text{–}4p, 12\text{+}2p)$

We consider two codes corresponding to p = 0 and p = 1. The codes can correct upto 5+p errors and 2–D decoding algorithm is of type–1. The conjugacy class leaders of the zero locations are given by (0,1), (0,3), (0,5), (0,7), (1,0) and (1,11–p) to (1,14). The configuration of zero locations is shown in Figure 6.4.4 for p = 0 and Figure 6.4.5 for p = 1.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z											Z	Z	Z	Z
2	Z							Z		Z		Z		Z	
3	Z						Z	Z						Z	Z
4	Z			Z				Z				Z			Z

Figure 6.4.4 Configuration of zeros of (75, 41, 12) code

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z										Z	Z	Z	Z	Z
2	Z					Z		Z		Z		Z		Z	
3	Z					Z	Z	Z						Z	Z
4	Z			Z				Z			Z	Z			Z

Figure 6.4.5 Configuration of zeros of (75, 37, 14) code

Decoding Algorithm for $(75, 41-4p, 12+2p)$

1) Pass $S_{1,1}$ to $S_{1,14}$ through B–M algorithm and find $\Lambda(z)$. Determine the rest of syndromes of row–0 by recursion.

2) If $\deg(\Lambda(z)) \leq 3+p$, then update $\Lambda(z)$ by passing $S_{1,11-p}$ to and $S_{1,0}$ through B–M algorithm.

3) Compute the syndromes of row 1 by recursive extension using $\Lambda(z)$. Then determine the syndromes of the remaining rows using conjugacy property.

4) 2–D IDFT of S gives e.

6.4.5 Decoding of codes of form $(75, 32-4p, 16+2p)$

We consider 5 codes corresponding to $p = 0$ to 4. The codes have $e_{2-D} = 7+p$ and have type–1 decoding algorithm. The conjugacy class leaders of the zero locations are given by $(0,0)$, $(0,1)$, $(0,3)$, $(0,5)$, $(0,7)$, $(1,0)$, $(1,9-p)$ to $(1,14)$. The configuration of zeros is shown in Figure 6.4.6 for the case p=0.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z									Z	Z	Z	Z	Z	Z
2	Z			Z		Z		Z		Z		Z		Z	
3	Z					Z	Z	Z					Z	Z	Z
4	Z			Z			Z	Z			Z	Z			Z

Figure 6.4.6 Configuration of zeros of $(75, 32, 16)$ code $(p = 0)$

Some of these codes those corresponding to $p = 0$ to 2, have the maximum possible d_{min} among the linear codes with the same length and dimension. The code corresponding to $p = 4$ $(75, 16, 24)$ is not given in the Table of [56]. But the table lists the code $(75, 17, 24)$ which has one more information bit than $(75, 16, 24)$ code. As the decoding algorithm for $(75, 17, 24)$ turns out to be complex and the gain is just a single information bit, we do not give the decoding algorithm for the latter.

Decoding Algorithm for $(75, 32-4p, 16+2p)$

1) Compute parity checks for all columns. Determine $\Lambda(z)$ using locators of columns with odd parity.

2) If $\deg(\Lambda(z)) \leq 5+p$, then update $\Lambda(z)$ by passing $S_{1,9-p}$ to $S_{1,0}$ through B–M algorithm.

3) Compute the syndromes of row 1 by recursive extension using $\Lambda(z)$. Then determine the syndromes of the remaining rows using conjugacy constraints.

4) 2–D IDFT of S gives e.

6.4.6 Decoding of (75, 12, 28)

This code has $e_{2-D} = 13$ and 2–D decoding is of type–3. The conjugacy class leaders of the zero locations are given by (0,0), (0,1), (0,3), (0,5), (0,7), (1,3), (1,5), (1,7), (2,3), (2,5), (2,7), (3,1), (3,3), (3,7), (4,1), (4,3), and (4,7). The configuration of zero locations is as shown in Figure 6.4.7.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1		Z	Z	Z	Z	Z			Z	Z	Z	Z	Z	Z	Z
2			Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
3		Z	Z	Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
4		Z		Z		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Figure 6.4.7 Configuration of zeros of (75, 12, 28)

Derivation Strategy : The first stage consists of row decoding based on (15, 4, 8) code. Assign correction numbers to rows which are corrected. Form $\phi_{uc}(z)$ with uncorrectable row locators as roots.

In the second stage decoding, at most 3 columns could be in error. First $\phi(z)$ is determined from 3 syndromes of column–1 with $\phi_{uc}(z)$ as the initialization polynomial. If $\phi(z)$ is not valid, then 3 least reliable columns are used for recursive extension. Using $\phi(z)$ syndromes of all the columns except column–0 are found. Before proceeding further it is necessary to remove errors in incorrectly decoded rows introduced in the first stage of decoding. Then $\Lambda(z)$ is computation is straightforward and the remaining syndromes are computed.

Decoding Algorithm for (75, 12, 28)

1) Compute 1–D DFT of r to obtain $R'_{i,j}$ for rows $i = 0$ to 6, at locations $j\epsilon \{9, 10,..., 14\}$. Process $\{R'_{i,j}\}$ based on (15, 5, 7) code. The ith row, if corrected, assigned w_i equal to number of errors corrected. Form $\phi_{uc}(z)$ from locators of rows which are detected as uncorrectable.

2) Find the 1–D DFT of columns of R' to obtain S.

3) If $\deg(\phi_{uc}(z)) \leq 1$, compute $\phi(z)$ by passing $\{S_{3,1}, S_{4,1}, S_{0,1}\}$ through B–M algorithm with $\phi_{uc}(z)$ as the initialization polynomial and check for validity. If $\phi(z)$ is not valid, then $\phi(z)$ is formed from locators of 3 least reliable decoded columns.

4) Determine all the syndromes of columns 1, 2, 4 and 8 using $\phi(z)$.

5) If there are incorrectly decoded rows, remove errors introduced in step 1 and adapt the syndromes.

6)   Compute parity checks for all columns and form $\Lambda(z)$ from the locators of columns with odd parity. If $\deg(\Lambda(z)) \leq 11$, update $\Lambda(z)$ by passing through $S_{1,1}$ to $S_{1,14}$. Compute $S_{1,0}$ to $S_{4,0}$ using $\Lambda(z)$ by row recursion.

7)   2–D IDFT of S gives e.

6.4.7 Decoding of $(75, 9, 32)$

This code has $e_{2-D} = 15$ and 2–D decoding is of type–3. The conjugacy class leaders of the zero locations are given by $(0,1)$, $(0,3)$, $(0,5)$, $(0,7)$, $(1,0)$, $(1,3)$, $(1,5)$, $(1,7)$, $(2,3)$, $(2,5)$, $(2,7)$, $(3,1)$, $(3,3)$, $(3,7)$, $(4,1)$, $(4,3)$, and $(4,7)$. The configuration of zero locations is as shown in Figure 6.4.8.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
1	Z		Z	Z	Z	Z	Z	Z		Z	Z	Z	Z	Z	Z
2	Z			Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
3	Z	Z	Z	Z		Z	Z	Z		Z	Z	Z	Z	Z	Z
4	Z	Z		Z		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Figure 6.4.8 Configuration of zeros of $(75, 9, 32)$

Derivation Strategy : The first stage consists of row decoding based on $(15, 5, 7)$ code. Assign correction numbers to rows which are corrected. Form $\phi_{uc}(z)$ with uncorrectable row locators as roots.

In second stage at most 3 columns could be in error. First $\phi(z)$ is determined from 3 syndromes of column–1 with $\phi_{uc}(z)$ as the initialization polynomial. If $\phi(z)$ is not valid, then 3 least reliable decoded columns are used for recursive extension. For this code, finding $\phi(z)$ is enough to complete recursive extension.

Decoding Algorithm for $(75, 9, 32)$

1)   Compute 1–D DFT of the rows of r to obtain $R'_{i,j}$ for rows $i = 0$ to 6, at locations $j \epsilon \{9, 10, ...14,\}$. Process these syndromes based on decoding of $(15, 5, 7)$ code. The ith column, if corrected, assigned w_i equal to number of errors corrected. Form $\phi_{uc}(z)$ from the locators of columns which are detected as uncorrectable.

2)   Compute 1–D DFT of columns of R' to obtain S.

3)   Find $\phi(z)$ by passing $S_{1,0}$ to $S_{4,0}$ through B–M algorithm with $\phi_{uc}(z)$ as the initialization polynomial. If $\deg(\phi(z)) \leq 1$, update by passing through column–1 syndromes. $\phi(z)$ is checked for validity. If $\phi(z)$ is not valid, then $\phi(z)$ is formed from locators of the 3 least reliable columns.

4) Determine the remaining syndromes of columns 0, 1, 2, 4 and 8 using $\phi(z)$.

5) 2–D IDFT of S gives e.

Example 6.4.2: The transmitted code vector is given by /0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 /0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 / 0 0 0 0 1 0 1 0 0 1 1 0 1 1 / 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0/ 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1. The $e(x,y)$ is given by $((1+y+y^3+y^6) +x(y^2+y^5+y^7+y^{14}) +x^2(y^4+y^8+y^9) +x^3(y^{10}+y^{11}+y^{12}+y^{13}))$. The syndromes of R' that are to be processed are as shown below.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				α^7		α^{10}	α^{14}	α^{14}		α^{11}	α^5	α^7	α^{13}	α^{11}	α^{13}
1	1			α^{11}		1	α^7	α^{14}		α^{13}	1	α^7	α^{14}	α^{11}	α^{13}
2	0			α^9		0	α^3	α^7		α^{12}	0	α^{11}	α^6	α^{13}	α^{14}
3	1			α^{12}		α^5	α^9	α^7		α^6	α^{10}	α^{11}	α^3	α^{13}	α^{14}
4	1			0		0	0	0		0	0	0	0	0	0

Based on decoding of (15, 5, 7) row–0 and row–3 are detected as uncorrectable. Therefore $\phi_{uc}(z) = (1 +\alpha^7 z +\alpha^9 z^2)$. Row–1 has 4 errors, yet its syndromes satisfy $\Lambda_1(z) = (1 +\alpha^7 z +\alpha^4 z^2 +\alpha^6 z^3)$ and therefore is incorrectly decoded. Rows 2 and 4 are properly decoded. At step 3, B–M algorithm operating on $S_{0,1}$ to $S_{0,4}$ finds $\phi(z) = (1+\alpha^4 z+\alpha^{13} z^2+\alpha^{12} z^3)$. The bold entries show syndromes available and other entries are found using $\phi(z)$ by recursive extension.

r/c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	α^{10}	α^5	α^2	α^{10}	1	α^4	α	α^5	α	1	α^8	α^8	α^4	α^2
1	α^3	α	α^{10}	α^{10}	α^{11}	α^{11}	1	α^7	α^{14}	α^{12}	α^8	α^{13}	α	α^8	α^3
2	α^6	α^{13}	α^2	α^9	α^5	α	α^5	α^{11}	α^7	α^2	α^7	α	1	α^6	α^{14}
3	α^9	α^5	α^{13}	1	α^7	α^4	α^8	α^9	α^8	α^5	α^{13}	α^{11}	α^6	α^{14}	α^4
4	α^{12}	α^{14}	α^{11}	α^4	α^4	α^{14}	α^3	α^2	α^{10}	1	α^8	α^{12}	α^{10}	α^{13}	α^7

CHAPTER 7

CONCLUSION

The main thrust of the thesis has been to expand the range of codes which can be easily decoded using spectral techniques. Two classes of decoding techniques based on DFT and WHT transforms are studied in detail. Using these transforms the decoding problems, in the spectral domain, are shown to be equivalent to appropriate deconvolution problems. This viewpoint provides important insights into the nature of various decoding problems in the spectral domain. Structural properties of transforms and codes are utilized to obtain various decoding algorithms which, in essence. are appropriate procedures for performing specific deconvolutions. These procedures provide a broad categorizations of various decoding methodologies for RM and cyclic codes. WHT techniques are applied for decoding of RM codes of order 2 and 3, whereas DFT techniques are applied for decoding of different classes of cyclic codes.

7.1 Main results

7.1.1 WHT domain decoding algorithms

A mapping from binary {0,1} to real {−1,1} is necessary in order to apply WHT techniques. The received spectrum can be considered in the WHT domain as dyadic convolution of the error spectrum and the transmitted codeword spectrum. The error spectrum can accordingly be visualized as the impulse response of a linear filter which convolutionally distorts the transmitted codeword spectrum. The decoding problem can then be formulated as deconvolution of the WHT of received signal to get back the codeword spectrum, given some knowledge about error. The solution to deconvolution is based on characterization of histogram of WHT spectrum of RM codes. The WHT decoding algorithm directly gives codeword spectrum estimate. It turns out that decoding process mainly requires simple operations such as comparison and substitution.

We have shown that WHT spectrum of $R(2,2^m)$ consists of $\lfloor m/2+1 \rfloor$ different histogram classes, j^{th} class containing 2^{2j} nonzero spectral locations having value $\pm 2^{m-j}$ for j = 0 to $\lfloor m/2 \rfloor$. We have made use of automorphism group property of RM codes and a theorem concerning canonical representation of multivariable polynomials over a binary field to prove histogram characterization theorems. Unfortunately histogram characterization could not not be extended to third and higher order RM codes primarily because of non−availability of appropriate canonical forms.

We have also discussed techniques to analyze the modifications in the histogram of WHT spectrum of a RM code due to channel errors. However, this computation is in general quite tedious except in case of small errors. Fortunately, it is found that the knowledge of range of variation of spectral values is helpful

for decoding purpose. The maximum possible range of variation of a spectral value is linearly related to number of errors occurred.

The possibility of recovering the histogram of the transmitted codeword spectrum from that of the received word spectrum is based on the following two conditions: first, the sign of a nonzero spectrum should not change due to error modification and second, different spectral values within a histogram class are far enough so that their range of variation are distinct. An analysis of these conditions shows that it is not possible to decode $R(2,2^m)$ in a straightforward manner, the exception being $R(2,2^4)$, which is the first nontrivial single–error correcting second order RM code of length 16. We give a decoding algorithm for single–error correcting $R(2,2^4)$ code and utilize it to develop a decoding algorithm for general $R(2,2^m)$.

In order to obtain WHT domain decoding algorithm for general $R(2,2^m)$, we utilize superimposition property of a RM code to decompose a larger RM code into smaller RM codes having efficient decoding algorithms, which can be combined to provide a decoding algorithm correcting upto full error–correcting capability of the larger RM code. Earlier Tokiwa et. al. have shown that this approach to RM decoding algorithm results in less delay [46]. We have proposed a different decomposition algorithm that reduces decoding of $R(2,2^m)$ to $(m-4)$ fold decoding of $R(1,2^{m-1})$ and a single decoding of $2^{m-4}*R(2,2^4)$. A WHT domain decoding algorithm for decoding of $2^{m-4}*R(2,2^4)$ is given on the lines of WHT domain decoding algorithm for $R(2,2^4)$. This new WHT domain decoding algorithm and the well–known WHT domain decoding algorithm for $R(1,2^m)$ are combined to provide decoding algorithm for $R(2,2^m)$. Two variations of the decoding algorithm for $R(2,2^m)$ have been given, one of which avoids the need for inverse WHT and also simplifies the process of recovering information bits.

We have shown in a similar way that decoding of $R(3,2^m)$ reduces to $(m-5)$ fold decoding of $R(2,2^{m-1})$ and a single decoding of $2^{m-5}*R(3,2^5)$. WHT domain decoding algorithms for $R(3,2^5)$ and $2^{m-5}*R(3,2^5)$ are given and utilized for developing a decoding algorithm for $R(3,2^m)$ by combining decoding algorithms for $2^{m-5}*R(3,2^5)$ and $R(2,2^{m-1})$.

We have given decoder architectures for WHT domain decoding algorithms for second and third order RM codes. An estimate of the decoding delay with these architectures is given. For $m \geq 8$, the decoding delay for WHT domain decoding algorithms is shown to be significantly less compared to decoding delay for TSKN algorithm, which in turn has smaller delay compared to other decoding algorithms for higher order RM codes.

7.1.2 DFT domain decoding algorithms

We have treated both 1–D and 2–D decoding algorithms in a single framework. Both 1–D and 2–D DFT characterize a cyclic code as having specified components identically equal to zero. The received spectrum at these locations directly gives the value of the error spectrum. The direct availability of partial information on error is in contrast to its non–availability in case of WHT. By introducing an auxiliary

vector which is zero at locations corresponding to error locations, and considering the pointwise product of error vector and auxiliary vector and transforming the pointwise product vector, a zero–output convolution involving the error spectrum is formed. DFT domain decoding is accordingly shown to be formulated as zero–output deconvolution with the partial knowledge of error spectrum. In the DFT domain the deconvolution takes the form of problem of finding the connection polynomial of least order LFSR generating the entire error spectrum sequence with the known error spectrum as the initial values of the sequence. The other convolving component, the DFT of the auxiliary vector, is also of importance in that its zeros are related to the locations of the error and hence known as the error–locator polynomial.

Analysis of DFT domain decoding algorithms reveals three important stages: the error spectrum computation stage, the error location stage and the error evaluation stage. Any DFT domain decoding algorithm centers around the error location stage which involves formulation and computation of error location polynomials. Any particular decoding approach is typified by the nature of the deconvolution corresponding to the nature of the convolution of the transform domain vectors of the corresponding error locator and error spectrum polynomials. For efficient computation, the error location stage requires some structure on the zero locations of the code; this also influences the error evaluation stage. Error evaluation stage can be completed in many ways and an appropriate choice is made so as to reduce the overall decoding complexity.

We have studied two distinct DFT domain decoding approaches : 2–D deconvolution approach based on Sakata's algorithm for 2–D LFSR synthesis [47]–[49] and 1–D deconvolution approach based on a 2–D linear complexity theorem. The latter approach is our contribution based on a 2–D extension of Blahut's 1–D linear complexity theorem [50].

In Sakata's approach the basic idea is the association of a nonzero error position (i,j) with a 2–D error–locator (α^i,β^j) corresponding to row and column indices of the error position. Then the 2–D convolution of the transform domain vectors of the corresponding error–locator and error spectrum polynomials is zero and therefore the decoding problem is formulated as a 2–D zero–output deconvolution. Sakata has given an efficient algorithm for 2–D LFSR synthesis [47]–[49] generalizing B–M algorithm for 1–D LFSR synthesis. It is to be noted that both the syndrome and LFSR are 2–D in nature. The error evaluation stage consists of either solving for the roots of the system of connection polynomials for individual error positions in the binary case or performing 2–D recursive extension of the known error spectrum to obtain the complete error spectrum.

We have proposed an alternative 1–D deconvolution approach to obtain DFT domain spectral decoding algorithms centered around a generalization of Blahut's 1–D complexity theorem for a linear recurring sequence. Complexity of a sequence is defined as the minimal order of LFSR generating the sequence. This notion of complexity is shown to play a very useful role in the formulation and design of decoding algorithms, especially in spectral decoding algorithms for 2–D cyclic codes.

We have introduced a class of Γ 2–D BCH codes that has less number of zeros compared to the class of 2–D BCH codes but has the same error–correcting capabilities. Although it is not possible to apply Blahut's decoding algorithm directly for decoding of Γ 2–D BCH codes, we show that improved Blahut's decoding algorithm can be used to correct upto the limit determined from complexity theorem arguments. We have also given alternative mixed spectral and time domain implementations of Blahut's original as well as modified decoding algorithms. A comparative study of hardware requirements and delays for various implementations is given. It is found that time domain implementation of B–M algorithm in 2–D case is not as advantageous as time domain implementation of B–M algorithm in 1–D case, because of necessity of having to compute either row–transform or column transform of the received signal.

We have also investigated an alternative approach to decode 2–D BCH codes on the basis of multiple LFSR synthesis algorithm proposed by Tzeng and Feng [52]–[53]. It is found that for the 2–D BCH code this approach can correct a few more burst errors than Blahut's algorithm. But for Γ 2–D BCH codes both Blahut's decoding algorithm and multiple LFSR synthesis based decoding algorithm have the same error–correcting capability.

We have considered a class of 2–D BCH codes from the perspective of only random error correction and have given a decoding algorithm correcting upto designed distance for these codes. This study intuitively helps in evaluating the requirements on the structure of zero locations for correction different types of random and burst errors. It is found that at least t^2 zero locations are required for t random error correction. At least $3t^2$ zero locations are required for t x t burst error correction. Increasing the number to $4t^2$ does not increase correcting capability although it may reduce decoding computation.

b) DFT domain spectral decoding algorithms for cyclic codes

We next applied 1–D deconvolution approach to obtain decoding algorithms for various 1–D and 2–D cyclic codes. The decoding algorithms for 1–D cyclic codes are based on 2–D DFT characterization of cyclic codes. Although these cyclic codes are not as structured as BCH codes, their performance is comparable to BCH codes, and hence good decoding algorithms for them are of practical interest. We hope that the given 1–D deconvolution approach provides basic principles which can be tailored for decoding of specific codes. Almost all of the decoding algorithms discussed in the thesis perform only error–correction. Some examples of decoding algorithms performing errors–and–erasures correction are also given, showing that our techniques can also be adapted to erasure correction. In particular, erasure bursts are easily handled by 2–D decoding algorithms.

The main idea in developing the decoding algorithms is to analyze the changes in the degree of the connection polynomials due to different configurations of error patterns of a given weight, upto some limit which should desirably coincide with the correcting limit $\lfloor (d_{min}-1)/2 \rfloor$ of the code. The decoding

algorithms proceed in stages anticipating various possible error configurations, finding appropriate connection polynomials, and testing for their validity. Therefore validity tests form integral part of 2–D decoding algorithms and include periodicity tests on error spectrum, testing for conjugacy constraints and checking for consistency of generated row (column) sequence with known error spectrum along that row (column). Therefore, compared to 1–D spectral decoding algorithms, 2–D spectral decoding algorithms have more flexibility in the choice of both connection polynomials and recursive syndrome extensions. The price that has to be paid for this flexibility is that decoding algorithm becomes more complex.

The main effort is to find the appropriate row or column connection polynomial for a given configuration of errors using all possible information about the error spectrum. This is motivated by the engineering considerations and helps in a better understanding of the algorithm. The approach may be seen to carefully balance the requirements of algebraic complexity and implementation complexity.

We have given a classification of 2–D spectral algorithms into type–1, type–2 and type–3 in increasing order of implementation and computation complexity. In a type–1 algorithm computation of connection polynomials and performing recursive extension is direct. In a type–2 decoding algorithm either estimating unknown syndrome or the connection polynomial is necessary which is computationally expensive. A type–3 decoding algorithm can be considered as a spectral domain counterpart of concatenated decoding algorithm. The decoding involves two stages of individual row or column decoding followed by 2–D decoding. The advantage of type–3 algorithm is that it is not tied to any method of encoding of concatenated code.

We have chosen several 1–D cyclic codes of lengths 21, 33, 35, 39, 45, 51 and 63 from those compiled in [55]. Decoding algorithms for these codes exhibit different techniques and all of them decode upto the error–correcting capability of codes. For some of these codes an algebraic decoding algorithm has been proposed for the first time. Most of codes chosen for study have type–1 decoding algorithm. We have identified a class of t–random error–correcting codes of area 3 x n and 5 x n, with 2t consecutive zeros in row–0 and t consecutive zeros in any other row, having similar decoding algorithm. The computational burden of type–2 decoding algorithms depends on the number of guesses for unknown syndrome. For type–2 decoding algorithms discussed it is possible to choose appropriate locations such that computational burden is reasonable. For some codes which are not correctable upto their full capability by methods of [10], it is possible to do by 2–D decoding methods (for example (33, 11, 11) code). There are also codes in [10] (for example decoding algorithm for (45, 23, 7) code in [10]) for which 2–D spectral methods do not have any simple algorithm. There are other cases where Blahut's method of decoding beyond BCH bound may be simpler [4]. These instances are pointers to positive and negative features of 2–D DFT domain decoding algorithms.

Some good 2-D cyclic codes have been compiled by Jorn Jensen [56]. They are good in the sense that they have best known d_{min} among comparable linear codes. We have considered almost all codes of length 45, 49 and 75 in [56] and given decoding algorithms for them. It is found that most codes of length 45 that we have discussed have type-1 decoding algorithm. For some codes of length 45 from [56] having type-3 decoding algorithm we could find codes having same parameters but simpler type-1 decoding algorithm. For length 49 codes of area 7 x 7, it is in general difficult to perform recursive extension and consequently most of decoding algorithms are of type-3. Almost all codes of length 75, area 5 x 15, have type-1 decoding algorithm. A few of these codes have adaptable rates and they can be decoded by the same decoding algorithm with minor modifications. Such codes may have applications in some hybrid ARQ-FEC schemes.

7.2 Suggestions for further research

In the light of NP-completeness of hard-decision decoding [57] problem, it is important to obtain transformations applicable to large class of codes to reformulate the decoding problem so as to be solvable in polynomial time. Our efforts may be seen as an attempt to enlarge the scope of codes having tractable decoding algorithms using DFT and WHT domain spectral techniques.

Based on the main theme of the thesis, we suggest the following related problems for further investigation

1)   We have seen that from the system theoretic viewpoint decoding can be interpreted as an appropriate deconvolution. In this context, it is possible to obtain different spectral characterizations and corresponding decoding algorithms for the same code. By mapping binary {0, 1} to real {1,-1}, we can think of characterizing a cyclic code in terms of complex field DFT. It will be interesting to investigate nature of complex field DFT characterization of cyclic codes and develop suitable decoding (deconvolution) algorithms for them. What are the advantages of this method of decoding cyclic codes compared to finite field DFT domain decoding of cyclic codes?

2)   We have utilized superimposition property in combination with WHT techniques to provide decoding algorithms for $R(2,2^m)$ and $R(3,2^m)$. In both cases, special WHT domain decoding algorithms for $2^{m-4}*R(2,2^4)$ and $2^{m-5}*R(3,2^5)$ are required. In turn these decoding algorithms are based on WHT decoding algorithm for $R(2,2^4)$ and $R(3,2^5)$. The problem is to investigate WHT decoding algorithms for $R(r,2^{r+2})$ in general and use them to obtain WHT domain decoding algorithm for r^{th} order RM codes $R(r,2^m)$ for general m.

3)   It is known that for a linear code at least $2t^2$ parity checks are required in order to correct t x t 2-D cyclic burst errors [81]. We have given a class of Γ 2-D BCH codes that require $3t^2$ zero locations for t x t error correction, t^2 zeros less than typical 2-D BCH code. So there is a lot of scope of improvement.

In [82], a class of optimal t x t correcting codes meeting this bound from time domain point of view are discussed. It will be interesting to look into the minimum configuration of zero locations required for t x t correction from the spectral viewpoint.

4)   Another problem is to expand the range of cyclic codes which can be decoded using 2–D spectral domain decoding techniques presented in the thesis. Identification of a general class of codes having same decoding algorithm is of considerable interest. A comparative study of complexities of various decoding algorithms for cyclic codes is an allied problem.

5)   We have used histogram characterization of WHT spectrum of $R(2,2^m)$ to obtain WHT domain decoding algorithms for them. Classification of Boolean functions is a topic of considerable interest on its own merits. An investigation can be undertaken to see if the results on histogram characterization of RM codes (which have very close links with Boolean functions) can be used to obtain simple and useful canonical forms for Boolean functions.

REFERENCES

1) W. W. Peterson and E. J. Weldon, Jr., "Error Correcting Codes," 3rd Edition, Cambridge, M.I.T., 1961; 1972;

2) R.E. Blahut, "Algebraic Codes in the frequency Domain", in 'Algebraic Coding Theory and Applications'–Ed. by G.Longo, Springer–Verlag, Wien Newyork, 1979, pp 448–494

3) R.E. Blahut, "Transform Techniques for Error Control codes," IBM. J. RES. Develop., Vol 23, no. 3, pp. 299–315, May 1979.

4) R.E. Blahut, "Theory and Practice of Error Control Codes," Reading, MA: Addison–Wesley Publishing Company, California, 1983.

5) R. R. Green, "A serial orthogonal decoder," JPL Space Programs Summary, vol IV, No. 37–39, pp 247–253, 1966

6) E. C. Posner, "Combinatorial Structure in planetary reconaissance," in Error Correction Codes, H. B. Mann, Ed., New York, Wiley, 1969, pp 15–46

7) F. J. MacWilliams and N. J. A. Sloane, "Theory of Error–Correcting Codes," North–Holland Publishing Company, New York, 1981

8) J. Conan, " A Transform Approach to Alternant and the Like Error Correcting Codes," Jour. of Information & Optimization Sciences, Vol. 9, No. 3, pp 317–330, 1987

9) M. Loeloeian, and J. Conan, "A Transform Approach to Goppa codes," *IEEE Trans. on Inform. Theory*, VOL. IT–33, No. 1, pp 105–115, Jan. 1987

10) P. Bours, J.C.M. Janssen, M. van Asperdt and H.C.A. van Tilborg, " Algebraic Decoding Beyond e_{BCH} of Some Binary Cyclic Codes, when $e > e_{BCH}$," *IEEE Trans. on Inform. Theory*, VOL IT–36, No. 1, pp 214–222, Jan. 1990

11) S. L. Hurst, D. M. Miller and J. C. Muzio, "Spectral Techniques in Digital Logic," Academic Press Inc. (London) Ltd., 1985

12) E. R. Berlekamp, Edit. "Key papers in The Development of Coding Theory," IEEE Press, New York, 1974

13) G. C. Clark, Jr., and J. B., Cain, "Error–Correction Coding for Digital Communications," Plenum Press, New York, 1981

14) C. E. Shannon, and W. Weaver, "Mathematical Theory of Communication," Univ. of Illinois Press, Urbana 1949

15) Norman Abramson, "Information Theory and Coding," McGraw–Hill, NewYork, 1963

16) R. W. Hamming, "Error detecting and error correcting codes," Bell Syst. Tech. J., vol. 29, pp 147–160, 1950

17) M.J. E. Golay, "Notes on digital coding," Proc. IRE vol 37, p 657, June 1949

18) D. E. Muller, "Application of Boolean Algebra to switching circuit design and to error detection." IRE Trans. Electron. Comput. vol EC–3, pp 6–12 , Sept. 1954

19) I. S. Reed, "A class of multiple–error–correcting codes and the decoding scheme," IRE Trans. Inform. Theory, vol IT–4, pp 38–49, 1954

20) D. Slepian, "A Class of binary signaling alphabets," Bell Syst. Tech. J., vol 35, pp 203–234, 1956

21) E. Prange, "Cyclic Error–Correcting Codes in Two symbols," AFCRC–TN–57–103, Air Force Cambridge Research Center, Cambridge, Mass. Sept., 1957

22) D.A. Huffman, "A Linear circuit Viewpoint on Error–Correcting Codes," IRE Trans., IT–2, pp 20–28, 1956

23) W. W. Peterson, "Encoding and Error–Correctin Procedures for the Bose–Chaudhuri Codes," IRE Trans., IT–6, pp 459–470, Sept. 1960

24) J. E. Meggit, "Error Correcting Codes for Correcting Bursts of Errors," IBM J. Research Develop., 4, pp 329–334, 1960

25) I. S. Reed and G. Solomo, "Polynomial codes over certain finite fields," J. Soc. Ind. Appl. Math vol 8, pp 300–304, June 1960

26) R. C. Bose and D. K. Ray–Chaudhuri, "On a Class of Error–Correcting Binary Group Codes," Inf. and Control, No. 3, pp 68–79 1960

27) A. Hocquenghem, "Codes correcteurs d'erreurs" Chiffres, No. 2, pp 147–156, 1959

28) D. Gorenstein and N. Zierler, "A Class of Error–Correcting Codes in p^m symbols," J. Soc. Ind. Appl. Math., vol. 9, pp 207–214, June 1961

29) H. F. Mattson and G. Solomon, "A New Treatment of Bose–Chaudhuri Codes," J. Soc. Ind. Appl. Math., vol. 9, pp 654–669, Dec. 1961

30) R.T. Chien, "Cyclic Decoding Procedures for Bose–Chaudhuri–Hocquenghem Codes," *IEEE Trans. on Inform. Theory*, VOL IT–10, pp 357–363, Oct. 1964

31) G. David Forney, Jr., "On Decoding BCH Codes," *IEEE Trans. on Inform. Theory*, VOL IT–11, pp 549–557, Oct 1965

32) E. R. Berlekamp, "Algebraic Coding Theory," New York: McGraw–Hill, 1968

33) J. L. Massey, "Shift–Register Synthesis and BCH decoding," *IEEE Trans. on Inform. Theory*, VOL IT–15, pp 122–127, Jan. 1969

34) J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier Series," Math. Computat., vol. 19, pp 297–301, 1965

35) J. M. Pollard, "The Fast Fourier Transform in a Finite Field," Mathematics of Computation, vl. 25, No. 114, pp 365–373, Apr. 1971

36) W. C. Gore, "Transmitting Binary Symbols with Reed–Solomon Codes," Proceedings of Princeton Conference on Information Sciences and systems, Princeton, NJ, pp 495–497, 1973

37) R. T. Chien and D. M. Choy, "Algebraic Generalization of BCH–Goppa–Helgert Codes," *IEEE Trans. on Inform. Theory*, VOL IT–21, pp 70–79, 1975

38) A. Lempel and S. Winograd, " A new approach to error–correcting codes," *IEEE Trans. on Inform. Theory*, VOL IT–23, pp 503–508, July 1977

39) R. W., Boyd, J. B., Cain, B., Holt, and D. L., LaBanca, "High speed concatenated decoding technique for satellite communications," Proc. Nat. Electron. Conf., pp. 332–337, 1978

40) E. R. Berlekamp, "Bit Serial Reed–Solomon encoders," *IEEE Trans. on Inform. Theory*, VOL IT–28, No. 6, Nov. 1982

41) T. K. Truong, L.J. Deutsch, I.S. Reed, I. S. Hsu, K. Wang, C.S. Yeh, "The VLSI design of a Reed–Solomon encoder using Berlekamp's bit–serial multiplier algorithm," Jet Propulsion Lab. Pasadena, CA, TDA, Rep. 42–77, pp 72–86, Aug 15, 1982

42) I.S. Hsu, I.S. Reed, T.K. Truong, L.J. Deutsch, K. Wang, C.S. Yeh, "The VLSI implementation of a Reed–Solomon encoder using Berlekamp's bit–serial multiplier algorithm," *IEEE Trans. on Computers*, VOL C–33, No. 10, pp 906–911, Oct. 1984

43) Berlekamp, "The Technology of Error correcting codes," *Proc. of IEEE*, VOl 68, No. 5, pp 564–593, May 1980

44) H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen and I. S. Reed, "A VLSI Design of a pipeline R–S Decoder," *IEEE Trans. on Computers*, VOL C–34, No. 5, pp 393–403, May 1985

45) H. M. Shao and I. S. Reed, "On the VLSI Design of a pipeline R–S Decoder using Systolic Arrays," *IEEE Trans. on Computers*, VOL C–37, No. 10, pp 1273–1280, Oct. 1988

46) K. Tokiwa, T. Sugimura, M. Kasahara and T. Namekawa, "New Decoding algorithms for Reed–Muller Codes," *IEEE Trans. on. Inform. Theory*, VOL IT–28, No. 5, pp 779–787, Sept. 1982

47) S. Sakata, "Finding a Minimal Set of Linear Recurring Relations Capable of Generating a Given Finite Two–Dimensional Array," J. Symbolic Computation, No. 5, pp 321–337, 1988

48) S. Sakata, 'Synthesis of two–dimensional linear feedback shift registers,' in "Applied Algebra, Algebraic Algorithms and Error–Correcting Codes: Proceedings, AAECC–5, Menorca 1987," (L. Huguet and A. Poli, Eds.) pp. 399–407, Springer, Berlin

49) S. Sakata, "Extension of the Berlekamp–Massey Algorithm to N Dimensions," Information and Computation, vol 84, pp 207–239, 1990

50) J. L. Massey and T. Schaub, "Linear Complexity in Coding Theory," in "Coding Theory and Applications," Ed. G. Cohen and P. G. Godelwski, Lect. Notes in Comp. Sci., No. 311, pp 19-32, 1988

51) P. Fitzpatrick and G. H. Norton, "Finding a Basis for the Characteristic Ideal of an n-dimensional Linear Recurring Sequence," *IEEE Trans. on Inform. Theory*, VOL IT-36, No. 6, pp 1480-1487, Nov. 1990

52) G. L. Feng and K. K. Tzeng, "A Generalized Euclidean Algorithm for Multisequence Shift Register Synthesis," *IEEE Trans. on Inform. Theory*, VOl IT-35, No. 3, pp 584-594, May 1989

53) G. L. Feng and K. K. Tzeng, "A Generalization of Berlekamp-Massey Algorithm for Multisequence Shift Register Synthesis with Applications to Decoding Cyclic Codes," *IEEE Trans on Inform. Theory*, VOl IT-37, No. 5, pp 1274-1287, Sept. 1991

54) H. Shahri and K. K. Tzeng, " On Error-and-Erasure Decoding of Cyclic Codes," *IEEE Trans on Inform. Theory*, VOL IT-38, No. 2, pp 489-496, Mar 1992

55) J. H. Van Lint and R. M. Wilson, "On the Minimum Distance of Cyclic Codes," *IEEE Trans on Inform. Theory*, VOL. IT-32, No. 1, pp 23-40, Jan. 1986

56) Jorn M. Jensen, "The Concatenated Structure of Cyclic and Abelian Codes," *IEEE Trans. on Inform Theory*, VOL IT-31, No. 6, pp 788-793, Nov. 1985

57) E. R. Berlekamp, R. J. McEliece and H. C. A. van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Trans. on Inform. Theory*, VOL IT-24, No. 3, May 1978

58) M. U. Siddiqi and V. P. Sinha, "Signals and Systems Over Finite Groups and Monoids," in 'Systems and Signal Processing' Eds. R. N. Madan, N. Viswanadham and R. L. Kashyap, Oxford & IBH Publishing Co. Pvt. Ltd., New Delhi, pp 363-385, 1991

59) B. Sunder Rajan, "Spectral Characterization of Abelian Group Codes," M.Tech. Thesis, Dept. of Elec. Engg., IIT Kanpur, 1984

60) A. V. Oppenheim and R. W. Schafer, "Digital Signal Processing," Prentice-Hall of India Pvt. Ltd., New Delhi, 1988

61) E. J. Weldon, Jr., "Some results on majority-logic decoding," in H. B. Mann, ed., "Error-Correcting Codes," pp 149-162, John Wiley, New York, 1969

62) C. L. Chen, "Note on majority-logic decoding of finite geometry codes," *IEEE Trans. on Inform Theory*, VOL IT-18, pp 539-541, 1972

63) L. D. Rudolph and C. R. P. Hartmann, "Decoding by sequential code reduction," *IEEE Trans on Inform Theory*, VOL IT-19, pp 549-555, 1973

64) Simon N. Litsyn, "Fast Algorithms for decoding orthogonal and Related codes," Lect. Notes in Comp Sci., vol ,1990

65) G. Seroussi and A. Lempel, "Maximum likelihood decoding of certain Reed–Muller codes," *IEEE Trans. on Inform. Theory*, VOL IT–29, pp 448–450, 1983

66) Y. Be'ery and Jakov Snyders, "Optimal Soft–decision Block Decoder Based on Fast Hadamard Transform," *IEEE Trans. on Inform. Theory*, VOL IT–32, No.3, pp 355–364, May 1986

67) C. R. P. Hartmann and K. K. Tzeng, "Generalizations of the BCH bound," Information and Control., vol. 20, pp 489–498, 1972

68) C. Roos, "A new lower bound for the minimum distance of a cyclic code," *IEEE Trans. on Inform. Theory*, VOL IT–29, No. 3, pp 330–332, May 1983

69) Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa, "A method for solving key equation for decoding Goppa codes," Information and Control, vol. 27, pp 87–99, Jan. 1975

70) D. M. Mandelbaum, "A method for decoding of generalized Goppa codes," *IEEE Trans. on Inform. Theory*, VOL IT–23, pp 137–140, Jan 1977

71) Udaya. P. "Linear Feedback Shift Register Synthesis for Sequences over finite fields and Rings," MTech. Thesis, Dept. of Elec. Engg., IIT Kanpur 1987

72) J. L. Dornstetter, "On the Equivalence Between Berlekamp's and Euclid's Algorithms," *IEEE Trans. on Inform. Theory*, VOL IT–33, No. 3, May 1987

73) G. L. Feng and K. K. Tzeng "Decoding Cyclic and BCH Codes up to Actual Minimum Distance Using Nonrecurrent Syndrome Dependence Relations," *IEEE Trans. on Inform. Theory*, VOl IT–37, No. 6, pp 1716–1723, Nov. 1991

74) Rudolf Lidl and Harald Niederreiter, "Finite Fields" in Encyclopedia of Mathematics and its Applications. Vol. 20, Cambridge University Press, 1984

75) S. Sakata, "Decoding Binary 2–D Cyclic Codes by the 2–D Berlekamp–Massey Algorithm," *IEEE Trans. on Inform. Theory*, VOL IT–37, No. 4, pp 1200–1203, July 1991

76) B. Buchberger, Groebner bases; An algorithmic method in polynomial ideal theory," in 'Multidimensional Systems Theory,' N. K. Bose, Ed., Dordrecht: Reidel, pp 184–232, 1985.

77) Hurst S. L., "The Logical Processing of Digital Signals," Edward Arnold, London, 1978.

78) M. Y. Hsiao, "A class of optimal minimum odd–weight–column SEC–DED codes," IBM J. Res. Develop., vol. 14, pp. 395–401, 1970

79) C. L. Chen, "On majority–logic decoding of finite geometry codes," *IEEE Trans. on Inform. Theory*, VOL IT–18, pp 332–336, 1971

80) Chien R.T. and Ng. W., "Dual Product Codes for correction of Multiple low–density Burst errors," *IEEE Trans. Inform. Theory*, VOL. IT–19, No. 5, pp 672–677, Sept. 1973

81) H. Imai, "Two–dimensional burst–correcting codes," Trans. Inst. Electron. Commn. Eng. Japan, Vol.55–A, pp 9–18, August 1972

82) K. A. S. Abdel–Ghaffar, R.J. McEliece, H. C. A. Van Tilborg, "Two–dimensional Burst Identification Codes and Their Use in Burst Correction," *IEEE Trans. on Inform. Theory*, VOL. IT–34, No. 3, pp 494–504, May 1988

83) I.M. Boyarinov, "Method of decoding direct sums of Products of codes and its applications," Prob. Pere. Informatsii, Vol 17, No 2, pp 39–51, April–June 1981.

84) V.A. Zinov'ev and V.V. Zyablov, "Correction of error bursts and independent errors using generalized cascade codes," Vol. 15, No. 2, pp 58–70, April–June, 1979

85) R. L. Miller, "Minimal Codes in Abelian Group Algebras," Journal of Combinatorial Theory, Series A26, pp 166–178, 1979

86) K. Hwang, "Computer Arithmetic–Principles, Architecture and Design," New York: John Wiley, 1979

87) Madhusudhana H. S. and M. U. Siddiqi, "On Blahut's decoding algorithm for two–dimensional BCH codes," pp 94, Proceedings, 1991 IEEE International Symposium on Information Theory, Budapest June 24–28, 1991

88) I. Niven and H. S. Zuckerman, "An Introduction to the Theory of Numbers," Wiley Eastern, 1976 .